APPLE II
REFERANCE MANUAL

APPLE II
系統介紹與使用手册



電腦技術社印行

APPLE II
系統介紹與使用手册

電腦技術社印行

APPLE II 系統介紹與使用手册

出版：電　腦　出　版　社
香港北角渣華街19號A 地下

印刷：藝　力　印　刷　公　司
香港柴灣新安街4 號12樓D

1983年3月出版　定價港幣25元正

目　錄

第一章　瞭解您的APPLE微電腦 …………………………… 7

一、電源供應器 …………………………… 7

二、主電路板 …………………………… 8

三、如何與您的APPLE交談 …………………………… 9

四、鍵盤 …………………………… 10

五、熟悉鍵盤 …………………………… 11

六、APPLE螢光幕顯示器 …………………………… 14

七、螢光幕連接器 …………………………… 14

八、歐洲型APPLE變更器 …………………………… 15

九、螢光幕的格式 …………………………… 17

十、螢光幕的記憶區 …………………………… 18

十一、螢光幕的記憶頁 …………………………… 18

十二、螢光幕的開關 …………………………… 19

十三、主要編輯畫面的形式 …………………………… 20

十四、低解析度繪圖形式 …………………………… 23

十五、高解析度繪圖形式 …………………………… 26

十六、其他的輸入／輸出情況 …………………………… 28

十七、APPLE的一些差異 …………………………… 34

第二章　與APPLE交談 …………………………… 38

一、標準的輸出 …………………………… 38

二、標準的輸入 …………………………… 42

三、脫離狀況的代碼…………………………………… 45
四、重定週期 ………………………………………… 47

第三章　系統監督程式……………………………… 52

一、進入系統監督程式……………………………… 52
二、位址與資料的使用……………………………… 53
三、查詢記憶體的內容……………………………… 54
四、改變記憶體位置的內容………………………… 60
五、更改一連串位址的內容………………………… 61
六、記憶體區域的搬移……………………………… 62
七、兩個記憶體區域的比較………………………… 65
八、存放記憶體的資料到磁帶機上………………… 66
九、讀取磁帶上的資料……………………………… 68
十、機器語言………………………………………… 69
十一、組合語言……………………………………… 72
十二、程式的除錯…………………………………… 75
十三、檢查及更改暫存器…………………………… 78
十四、其他的MONITOR命令 ……………………… 80
十五、MONITOR的特殊用法………………………… 82
十六、增加MONITOR的命令………………………… 86

第四章　記憶體的組織……………………………… 97
一、隨機取存記憶體的儲存區……………………… 98
二、隨機取存記憶體的結構………………………… 101
三、唯讀記憶體的儲存區…………………………… 104

四、輸入／輸出的位置……………………………… 106

第五章　輸入及輸出之結構……………………………… 109

一、固定的輸入與輸出……………………………… 109
二、週邊面板的輸入／輸出……………………………… 113
三、週邊面板的輸入／輸出空間……………………………… 113
四、週邊面板的ROM空間 ……………………………… 114
五、設計輸入／輸出程式時的要點……………………………… 115
六、週邊設備插座的SCRATCHPAD RAM ……………… 118
七、CSW／KSW輸出開關 ……………………………… 120
八、擴充唯讀記憶體……………………………… 121

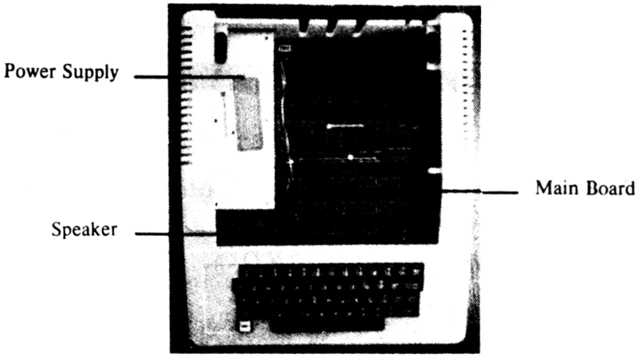
第六章　硬體的外形結構……………………………… 124

一、微處理機的簡介……………………………… 124
二、系統時序……………………………… 126
三、電源供應器……………………………… 128
四、唯讀記憶體……………………………… 130
五、隨機取存記憶體……………………………… 131
六、螢光幕產生器……………………………… 133
七、螢光幕輸出插孔……………………………… 135
八、固定的輸入與輸出……………………………… 136
九、“USER 1”JUMPER ……………………………… 138
十、遊戲的輸入與輸出連接器……………………………… 139
十一、鍵盤……………………………… 140
十二、鍵盤的連接器……………………………… 142
十三、卡式磁帶機介面板的插孔……………………………… 143

十四、電源連接器……………………………………………… 143
十五、發音器……………………………………………………… 144
十六、週邊設備連接器………………………………………… 145
附　　錄…………………………………………………………… 157
附錄A　THE 6502 INSTRUCTIONS SET………… 158
附錄B　SPECIAL LOCATIONS …………………… 170
附錄C　ROM LISTING ……………………………… 176
附錄D　GETTING APPLESOFT BASIC UP
AND RUNNING ……………………………… 219
附錄E　名詞解釋………………………………………………… 223

第一章　瞭解您的APPLE微電腦

對您的Apple 微電腦若需要更完整的資料介紹的話，請參看“Apple BASIC 程式語言與操作”一書中之第一、二章。在本書中所提到的Apple 都是以鍵盤朝著您的方向爲正確方向，當您想打開蓋子時，需用雙手輕輕地推起機器後面上緣的板子，直到聽到“喀”聲時才可將板子拉起，這時您將看到：



照片一　The Apple Ⅱ

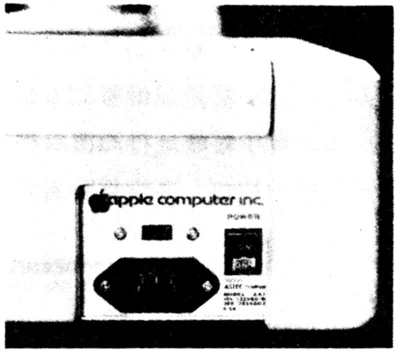
一、電源供應器（THE POWER SUPPLY）

在機器左方有一金屬盒子，此即爲電源供應器。它提供了四種電壓：+5V，−5.2V，+11.8V，−12.0V。它爲一高頻式電源供應器，具有避免在兩種輸出的輸出電壓時產生不平衡之現象。電源綫只要直接插入後方之插座即可，而此電源供應器之開關在它自己裏面

當我們開機時，它裏面的開關才會打開而供應電源，如此可以避免我們開機時有觸電的危險。



110 volt model



110/220 volt model

照片二　The back of the Apple Power Supply

二、主電路板（ THE MAIN BOARD ）

此板有兩種模式，即原板（ Revision 0 ）和修訂版（ Revision 1 ），這兩種板只是在一些電路上有點差異而已。

在這板上有大約 80 個 IC（ 積體電路 ）和少量的其他組件。在板子的中央，也就是 8 個長形的插座（ slot ）之前方有一個比其他的要大的 IC，這個 IC 即爲 Apple 電腦的大腦，它就是 6502 微處理機（ microprocess ）。它每秒可以做 1,023,000 次循環（ machine cycles ），也就是說它每秒可以處理 50 萬次加法或減法。它可以容納的位置之最大極限爲 65,536 位元組（ byte, 1 byte = 8 bits ），而在其中的 13 個位置上包含了 56 個指令。此微處理機也用在其他的電腦上，或是其他的電子設備上。在微處理機之下方有 6 個插孔，上面也許會插入 1 到 6 個的小 IC，這些 IC 叫做唯讀記憶體（ ROM ）

，在這些 ROM 裏面存入了程式，當我們一開啟機器時就可使用這些程式，這些程式包括Apple 系統監督程式（ Apple System Monitor ），Apple 自動啟動程式（ Apple Autostart Monitor ）， Apple Integer BASIC 及Applesoft Ⅱ BASIC 和Apple之一些庫存副程式。您的Apple 是否擁有這些呢？這要看您有沒有購買這些ROM了。

在ROM之右下方，用白線圈起之長方形內，有 24 個 IC 插孔。也許有一些或全部都插上了 IC，這就是Apple 之隨機取存記憶體（RAM ），也就是我們通常所稱之記憶體，在這三列的插孔上，可以放置 4,096 到 49,152 位元組（ 4K～ 48K ） 的記憶體，亦即每一列上可以放置 4K 到 16K的記憶體。每列上的 8 個 IC 須爲同一種型式之 IC，但在不同列上我們可以插入不同型式之 IC，此種 IC 可以有 9 種不同之組合。 RAM 記憶體的目的是要存入程式或是資料，但是在電腦關機後，這些程式與資料就全部自動的消失了。

在Apple 主電路板上之其他組件有各種不同之功能：它們控制著資料之傳遞或讀入資料，或把資料送到顯像螢光幕上，或利用聲音傳遞出來。

在Apple 主電路板上方有 8 個週邊設備插座，每一個插座上均可插入一面電路板，可以增大 Apple 的RAM或ROM記憶體，或接上輸入及輸出的設備，這些插座通常稱爲Apple 之母板。

三、如何與您的 Apple 交談

要與Apple 交談就得全靠我們的雙手了。各種程式，各種語言要與Apple 交談那就只有用鍵盤了，它有點像打字機的鍵盤，但它的安排及一些特別鍵跟一般的打字機有些不同。

我們用雙手來交談，所以也要用眼睛來看，我們打入的資料包括了文字、數字、符號和一些色彩，這些均將以黑白或彩色的顯像終端

機顯示出來。

四、鍵盤（THE KEYBOARD）

The Apple Keyboard

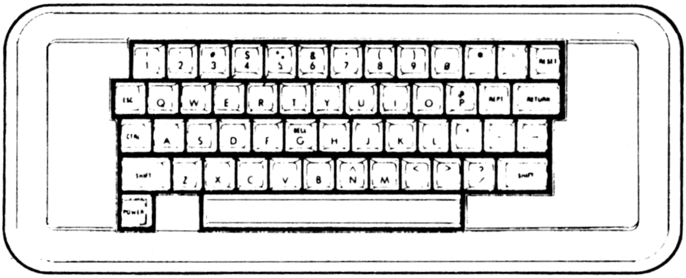
Number of Keys:	52
Coding:	Upper Case ASCII
Number of codes:	91
Output:	Seven bits, plus strobe
Power requirements:	+5v at 120mA −12v at 50mA
Rollover:	2 key
Special keys:	CTRL ESC RESET REPT ← →

Memory mapped locations:

	Hex	Decimal	
Data	$C000	49152	-16384
Clear	$C010	49168	-16368

Apple 鍵盤上有52個字鍵，當鍵入電腦時它們即轉換成美國國家標準碼（ASCII）而這 52 個字鍵可產生 91 種碼（用SHIFT 控制鍵），表二：即將這些鍵以ASCII 碼來表示，圖三：爲鍵盤的圖形。

鍵盤的信號是由一條有16根電線組合之電纜接到 Apple 之主電路板上一個 IC的 16 支釘脚，大多數Apple 可使用的程式語言都可以直接從鍵盤上讀入資料（如BASIC語言之 INPUT和GET 指令），當然我們的程式也可以直接由鍵盤打入。



照片三　Apple 的鍵盤

五、熟悉鍵盤

鍵盤將每一鍵入的信號以 7 個位元（ bit ）送入電腦裏面，而第 8 個位元爲一個信號開關。當字鍵按下去時，此開關即將 7 個位元資料讀入 Apple 之記憶位置上，因此執行程式時就會到此位置去找資料，而此位置之值爲 128 或更大，也就是說我們按的鍵會被轉換成數值來存放，表三即爲鍵入之ASCII 代碼被轉換成數值的形式，此數值將被保留到下一個鍵入的數值取代它，或是直到我們的程式告訴記憶位置不必再保留該數值。

一旦我們的程式已接收且知悉一個按入的鍵時，它將告訴鍵盤的記憶位置釋放這一個鍵，並且準備去接收下一個新的鍵，我們的程式可以如此做乃是利用另一個記憶位置的方法。當我們參用到其他位置的時候，在第一個位置的值將降到 128 以下，這一個值將保持低值直到我們按進另一個鍵，這一個動作稱爲“清除鍵盤特殊信號”。我們的程式能執行讀或寫到這--個特殊位置上，而被寫到這一個位置的資

料或是從這一個位置讀取的資料是不相關的，這就是唯一的清除鍵盤特殊信號的參用方式。一旦我們清除了鍵盤特殊信號，我們仍舊能找回這一個剛剛按進的鍵之代碼，其方式是加128（十六進位制的$80）到這一鍵盤位置上的值。

下列是一些由鍵盤使用的特殊記憶位置：

Table 1: Keyboard Special Locations			
Location Hex	Decimal		Description
$C000	49152	-16384	Keyboard Data
$C010	49168	-16368	Clear Keyboard Strobe

表　一

在鍵盤右上角的地方有一 RESET 鍵，這一個鍵並不產生ASCII 代碼。但除此之外，這鍵是直接與微處理機連接的，當這 RESET 鍵被按下時，所有的處理均停止；當這鍵被放開時，則電腦開始一個重新設定的廻路。在後面我們將對 RESET 加以詳細說明。

CTRL 與 SHIFT 鍵，本身亦不產生代碼，但是會改變其他鍵所產生的代碼。 REPT 鍵如果被單獨按下時，則將上次已產生的代碼予以重複地產生；如果我們按著某個字鍵不放，然後再按 REPT 鍵不放，那麼就如同我們重複的按該字鍵一樣，而且是以每秒鐘按十次字鍵的速度重複著。當我們放掉該字鍵或是 REPT 鍵時，這一重複的情況立即結束。

在鍵盤左下角的一個POWER燈是一個指示燈，它是用來表示這Apple 的電源開關是否已經打開。

Table 2: Keys and Their Associated ASCII Codes

Key	Alone	CTRL	SHIFT	Both	Key	Alone	CTRL	SHIFT	Both
space	$A0	$A0	$A0	$A0	RETURN	$8D	$8D	$8D	$8D
0	$B0	$B0	$B0	$B0	G	$C7	$87	$C7	$87
1!	$B1	$B1	$A1	$A1	H	$C8	$88	$C8	$88
2"	$B2	$B2	$A2	$A2	I	$C9	$89	$C9	$89
3#	$B3	$B3	$A3	$A3	J	$CA	$8A	$CA	$8A
4$	$B4	$B4	$A4	$A4	K	$CB	$8B	$CB	$8B
5%	$B5	$B5	$A5	$A5	L	$CC	$8C	$CC	$8C
6&	$B6	$B6	$A6	$A6	M	$CD	$8D	$DD	$9D
7'	$B7	$B7	$A7	$A7	N^	$CE	$8E	$DE	$9E
8(	$B8	$B8	$A8	$A8	O	$CF	$8F	$CF	$8F
9)	$B9	$B9	$A9	$A9	P@	$D0	$90	$C0	$80
•	$BA	$BA	$AA	$AA	Q	$D1	$91	$D1	$91
;+	$BB	$BB	$AB	$AB	R	$D2	$92	$D2	$92
,<	$AC	$AC	$BC	$BC	S	$D3	$93	$D3	$93
-=	$AD	$AD	$BD	$BD	T	$D4	$94	$D4	$94
.>	$AE	$AE	$BE	$BE	U	$D5	$95	$D5	$95
/?	$AF	$AF	$BF	$BF	V	$D6	$96	$D6	$96
A	$C1	$81	$C1	$81	W	$D7	$97	$D7	$97
B	$C2	$82	$C2	$82	X	$D8	$98	$D8	$98
C	$C3	$83	$C3	$83	Y	$D9	$99	$D9	$99
D	$C4	$84	$C4	$84	Z	$DA	$9A	$DA	$9A
E	$C5	$85	$C5	$85	→	$88	$88	$88	$88
F	$C6	$86	$C6	$86	←	$95	$95	$95	$95
					ESC	$9B	$9B	$9B	$9B

表　　二

在表二裡所有的代碼皆以十六進位制表示，如果要對照十進位制的代碼請參照表三。

Table 3: The ASCII Character Set

Decimal	Hex	128 $80	144 $90	160 $A0	176 $B0	192 $C0	208 $D0	224 $E0	240 $F0
0	$0	nul	dle		0	@	P		p
1	$1	soh	dc1	!	1	A	Q	a	q
2	$2	stx	dc2	"	2	B	R	b	r
3	$3	etx	dc3	#	3	C	S	c	s
4	$4	eot	dc4	$	4	D	T	d	t
5	$5	enq	nak	%	5	E	U	e	u
6	$6	ack	syn	&	6	F	V	f	v
7	$7	bel	etb	'	7	G	W	g	w
8	$8	bs	can	(	8	H	X	h	x
9	$9	ht	em	)	9	I	Y	i	y
10	$A	lf	sub	•	:	J	Z	j	z
11	$B	vt	esc	+	;	K	[	k	{
12	$C	ff	fs	,	<	L	\	l	|
13	$D	cr	gs	—	=	M	]	m	}
14	$E	so	rs	.	>	N	^	n	~
15	$F	si	us	/	?	O	_	o	rub

表　　三

在第二及第三組的小寫字母是標準ASCII控制字元的縮寫。

並非所有在表中的文字皆能由鍵盤上產生，特別是在右邊的兩行（也就是小寫字母）及符號［，＼，—，以及控制字元“fs”,“us”“rub”，都無法在Apple的鍵盤上找到。

在上表中的任何文字，其十進位或是十六進位的值，即在最上列的十進位或十六進位數字與表中左邊的十進位或十六進位數字所表示相對位置上的那個字元。例如：控制字元“nul”的ASCII代碼爲128（或是十六進位制的$80），又控制字元“soh”的ASCII代碼爲129（或是十六進位制的$81）。

六、APPLE螢光幕顯示器（THE APPLE VIDEO DISPLAY）

顯示形態：	記憶體映到系統的RAM上。
顯示型式：	主要編輯畫面，低解析度繪圖，高解析度繪圖。
主要編輯容量：	960字元（24列，40行）。
字元形態：	5×7點矩陣。
字元型式：	Normal, Inverse, Flashing.
繪圖容量：	1920色塊（低解析度）由40×48的行列組成。
	53760點（高解析度）由280×192的行列組成。
色彩種類：	16種（低解析度繪圖）
	6種（高解析度繪圖）

七、螢光幕連接器（THE VIDEO CONNECTOR）

在Apple Ⅱ電路板上的右底角地方，有一個金屬的連接器，其標記爲“VIDEO”，這一個連接器允許我們接上一個電纜來連接Apple和一個封閉電路的螢光幕監督器，在連接的電纜上的一端必須有一個

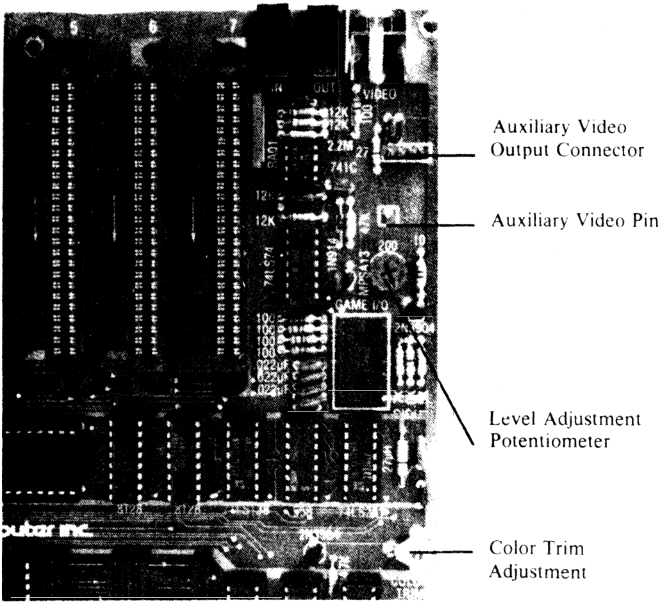
陽極的RCA插孔，以便插入Apple上，另一端則必須與我們所使用的特殊裝置互相一致的連接器。在Apple從連接器上發出的信號與標準EIA信號相似，這是一種經由國際電視標準委員會所共認與確定的彩色電視訊號，這一訊號的標準可從0V到1V的尖峯來調整，調整器在電路板上的右邊，大約離主電路板後面3吋的地方有一可轉動的小型電位計。

有一個不能調整的，具有2V的相同訊號可在另外兩個地方使用到：其中之一是在電路板後面左邊2吋的一個單一的包線釘脚，另外是電路板靠左後方的4個相同釘脚中的一個釘脚，這4個釘脚的另外3支釘脚是用來連接－5V，＋12V及地線的。

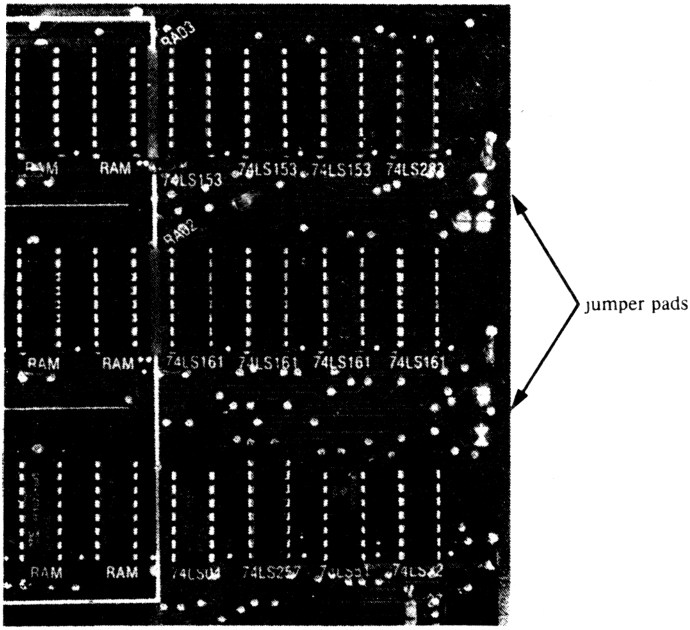
八、歐洲型APPLE變更器（EURAPPLE（50HZ）MODIFICATIO

你的APPLE能被加以改變後產生一種螢光幕信號，這一信號可與標準CCIR共用，這一標準在歐洲的許多國家中使用。要做這一個變更，只要割掉二個X—型的插孔（pads），這一插孔在APPLE電路板上的右邊離背部大約有9英吋的地方，同時把X型的插孔跟3個O形的插孔連接在一起，這三個O型的插孔在X型位置附近。這時你可把APPLE的螢光幕連接器接到歐洲標準型封閉電路，黑白或彩色電視機上。假如你願意，你能獲得一個"歐洲彩色"轉換器，轉換這螢光幕信號成爲PAL或SECAM 標準彩色螢光幕信號，適合任何歐洲電視接收器的使用。這一轉換器是一個很小的印刷電路板，插在最右邊的週邊插孔編號7上面，同時連接到信號輔助螢光幕輸出釘脚。

WARNING：這一個變更器在你的APPLE上保證是空的，同時需要置放不同的主要晶體，這一個變更器對初學者是不適合的。



照片四　螢光幕連接器與電流計



照片五　歐洲型（50HZ）Apple 的接續器插孔（Jumper Pads）

九、螢光幕的格式（SCREEN FORMAT）

對我們的Apple 所接的螢光幕顯示器而言，有三種不同的訊息可顯示在這上面：

(1)主要編輯畫面：Apple 可以顯示24行數的特殊符號，及大寫字母，在每一行可顯示 40 個字元，這一字元是由 -7×5 的點矩陣所構成，在字與字之間隔是一點的距離，行與行之間的距離也是一點的距離。

(2)低解析度繪圖：Apple 能表示 1920 個彩色方塊，在一個 48× 40 色塊的行列內。每一色塊的彩色可以由一組有16個彩色的組中選一個顏色，在每一色塊之間沒有空間，因此要是有二個相鄰的色塊看起來，就像只有一個大一點的色塊一樣。

(3)高解析度繪圖：Apple 同時也可顯示彩色點在一 192 × 280 點的矩陣內，這些點與構成編輯字元的點是相同的大小，在高解析度繪圖的形式上有 6 種顏色可獲得，它們是黑、白、紅、藍、綠、橙，在螢光幕上的點可能是黑、白，或者是一個色彩，雖然對每點而言並非每一種顏色都可獲得。

當Apple 顯示一個特別的形態的訊息在螢光幕上，那就是說在這一特別的"形式"上，因此，如果你看一個字和數字在螢光幕上，那你可以很有把握的確定你的Apple 是在主要編輯畫面的形式。相同的，如果你看到一個充滿複雜色彩的顯示，你的電腦也許在低解析度繪圖形態上。

對每一形態的繪圖螢光幕來講，你也能有 4 行的編輯字幕在每一螢光幕的底部。這 4 行，取代在低解析度繪圖形態 48× 40行列式中的底下 8 排而使成為 40 × 40行列式，在高精準度繪圖形態上，這 4 行則取代底部的 32 排的點，使成為 160 × 280 的矩陣。你能使用這

一個“混合形態”來聯合顯示編輯和繪圖形態，但是並沒有方法來同時顯示繪圖的二種形式。

十、螢光幕的記憶區（SCREEN MEMORY）

螢光幕顯示使用在系統的 RAM 記憶區的訊息去產生它的顯示情況，一個單一的記憶區位置的値，控制某個固定的東西出現在影幕上，這一東西也許是一個字元，二疊彩色的色塊，或是 7 個點的一行。在編輯及低解析度的繪圖形式上一個記憶區包含 1024 位置，這一記憶區被用來做爲螢光幕訊息的來源。編輯和低解析度繪圖形式分用這一記憶區。在高解析度繪圖形式下，必須有一個隔離的，大的區域（8192 個位置），因爲有很大數量的訊息，需要被顯示。這些記憶區通常被稱爲“頁”，被保留給高解析度繪圖的區域有時被稱爲“圖像暫存區”，因爲這區域通常被用來存放一個映像或是繪圖。

十一、螢光幕的記憶頁（SCREEN PAGES）

事實上，對二個解析度的繪圖形式而言，這裡有二個區域使這二個解析度繪圖可以繪出它的訊息，第一個區域是被叫做“主要頁”或是“PAGE 1”，這第二個區域是被叫做“次要頁”或“PAGE 2”，同時次要頁是與主要頁的大小相同緊跟在第一區的後面。這個次要頁對於儲存的映像或是你想要能夠立即被顯示出來的編輯資料，一個程式能使用這二頁區域，使程式執行起來很生動，它先顯示頁，在另一頁上繪製的同時會突然的擲出這一頁。

編輯以及低解析度的繪圖共用次要頁的記憶區，就如同它們共用主要頁的記憶區一樣。在上面所提到的任何兩種形式的混合，同樣也可在次要頁上獲得，但是沒有任何方法混合這主要頁及次要頁在同一螢光幕上。

Table 4: Video Display Memory Ranges					
Screen	Page	Begins at		Ends at	
		Hex	Decimal		
Text/Lo-Res	Primary	$400	1024	$7FF	2047
	Secondary	$800	2048	$BFF	3071
Hi-Res	Primary	$2000	8192	$3FFF	16383
	Secondary	$4000	16384	$5FFF	24575

表　　四

十二、螢光幕的開關（SCREEN SWITCHES）

在不同的形式，頁，及混合式之間決定以那種形式來作的這種設備，我們稱它作"軟體開關"（Soft Switch）。稱它們爲開關，是因爲它們有二個位置（例如：開或關，編輯或繪圖），同時它們被稱爲"軟體"是因爲它們是由電腦的軟體程式來控制。

一個程式能"開動"一個開關是利用參用這一開關的在記憶體上的特殊位置。由這一位置讀取資料，或是把資料寫到這一位置上去兩者並無相關，這是參用這一位置所包含的位址而去開動這一開關。

對於螢光幕的軟體開關設定的控制，在Apple 裡有 8 個特殊的記憶位置具有此作用。它們是由兩位置一對一對的組合，當你參用了一對中的一個位置，你就是把這一位置對應的形式打開了，同時另一位置的形式就是被關了。這幾對的位置如下：

Table 5: Screen Soft Switches			
Location: Hex	Decimal		Description:
$C050	49232	-16304	Display a GRAPHICS mode.
$C051	49233	-16303	Display TEXT mode.
$C052	49234	-16302	Display all TEXT or GRAPHICS.
$C053	49235	-16301	Mix TEXT and a GRAPHICS mode.*
$C054	49236	-16300	Display the Primary page (Page 1).
$C055	49237	-16299	Display the Secondary page (Page 2).
$C056	49238	-16298	Display LO-RES GRAPHICS mode.*
$C057	49239	-16297	Display HI-RES GRAPHICS mode.*

表　五

這裡有10種不同種類的開關組合：

Table 6: Screen Mode Combinations					
Primary Page			Secondary Page		
Screen	Switches		Screen	Switches	
All Text	$C054	$C051	All Text	$C055	$C051
All Lo-Res Graphics	$C054 $C052	$C056 $C050	All Lo-Res Graphics	$C055 $C052	$C056 $C050
All Hi-Res Graphics	$C054 $C052	$C057 $C050	All Hi-Res Graphics	$C055 $C052	$C057 $C050
Mixed Text and Lo-Res	$C054 $C053	$C056 $C050	Mixed Text and Lo-Res	$C055 $C053	$C056 $C050
Mixed Text and Hi-Res	$C054 $C053	$C057 $C050	Mixed Text and Hi-Res	$C055 $C053	$C057 $C050

表　六

要設定Apple進入上表的任一形式，一個程式只要參用與設定這個形式開關的相對應的記憶位置內的位址。機器語言程式必須使用上表中的十六進位位址，BASIC程式則須用PEEK′或POKE指令所相對等的十進位值。這些開關可以以任何順序開啟，然而，如果開關進入繪圖形式中的任一個，在最後才開啟編輯／繪圖的開關是有帮助的。其他在形式上的改變將是在編輯形式之後發生，而這一發生是你無法看到的。因此，當設定繪圖形式時，被完成的圖形螢光幕將立即的全部將其顯示出來。

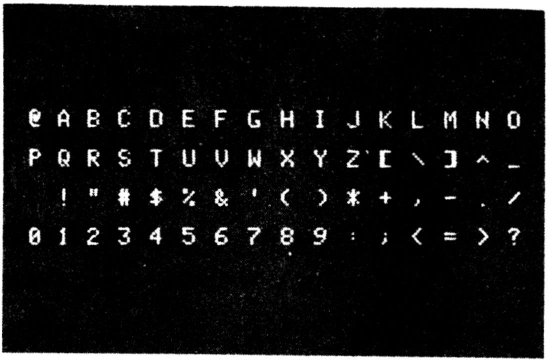
十三、主要編輯畫面的形式（THE TEXT MODE）

在編輯形式上，Apple可以顯示24行的字元，且每一行有40個字元，在螢光幕上的每一字元代表每一記憶體位置內的內容，而這一記憶位置是在被顯示的記憶頁內的記憶區。這些字元組包括了26個大寫的英文字母，10個阿拉伯數字，以及28個特殊符號，共有64個字元´。這些字元是點矩陣所組成，這一點矩陣是寬5點，高7個點共35點。每兩字元之間有一個點的距離以便分開相鄰的兩個字，同

時在上下行之間亦有一個點寬的距離，以便分開相鄰的兩行。字元的構成，在一般的情況是以白點在黑色的背景上構成。然而，每一在螢光幕上的字元亦可以黑點顯示在白色的背景上，或者二者方式不斷的交互改變，而產生一個閃動的文字。當這一螢光幕顯示是在編輯的形式下，在APPLE的螢光電路，將關掉送到電視監督器的彩色信號，而給你的是一個乾淨的黑白顯示幕。

被用來做爲主要編輯頁的記憶體區域，由位置1024開始，同時延伸到位置2047，其次的螢光幕是由位置2048開始，且延伸到位置3071。在機器語言方面，主要頁是由16進位的位址$400到位址$7FF，次要頁是由$800到$BFF。每一頁都有1024位元組的長度，對這1024個位元組而言，無論你再怎麼樣的做，在螢光幕上也只有960字元顯示在螢光幕上，剩下的64 個位元組沒有顯示在螢光幕上，這些位元組由儲存在Apple 的有智慧的介面週邊面板上的PROM裡的程式用來做爲暫存的位置。

照片6顯示在Apple 的螢光幕可獲得的64個字元。



照　　片　　六

表七，是表示上面 64 個字元的十進位制及十六進位制代碼。至於圖一則是Apple 的主要編輯畫面顯示圖及每一字元在螢光幕上位置

的記憶體位置之位址。

Table 7: ASCII Screen Characters

	Inverse				Flashing				Normal (Control)							(Lowercase)
Decimal	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
Hex	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
0 $0	@	P		0	@	P		0	@	P		0	@	P		0
1 $1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2 $2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	"	2
3 $3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4 $4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5 $5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6 $6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7 $7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7
8 $8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9 $9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10 $A	J	Z	*		J	Z	*		J	Z	*		J	Z	*	
11 $B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12 $C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13 $D	M	]	-	=	M	]	-	=	M	]	-	=	M	]	-	=
14 $E	N	^		>	N	^		>	N	^		>	N	^		>
15 $F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	/	?

表　　七

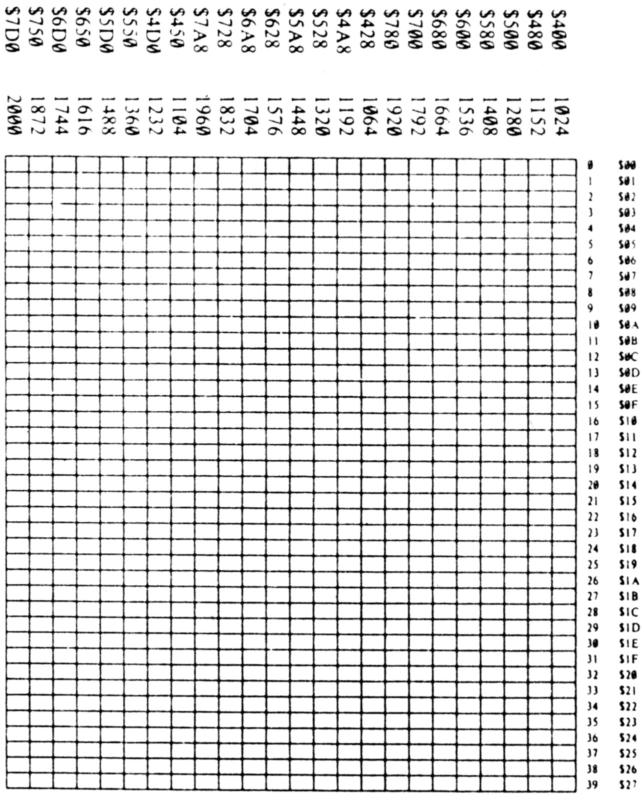


圖　一

十四、低解析度繪圖形式（THE LOW-RESOLUTION GRAPHICS MODE）

雖然是在低解析度的繪圖形式下，但Apple 仍然是以編輯的形式表現這 1024 個記憶體位置的內容，所不同的是以不同格式表現，在

低解析繪圖形式每一記憶位元組被顯示出來，並不是以ASCII字元表示，而是以二個彩色的色塊把一個疊在另一個上面的方式表示。螢光幕可以顯示一個 40 × 48的行列式色塊，每一色塊可能是16個顏色中的任一個，在一個黑白的電視機上，色彩是以灰色和白色的點出現。

對低解析度繪圖而言，由於在記憶頁上的每一位元組，是以二個塊（ blocks ）表現在螢光幕上，而且是垂直的疊放著，每一位元組被分成二個相等的部份，每一部份稱為“尼柏”（ Nybbles ），每一尼柏能保有一個值從0到15。在位元組的低尼柏上的值，決定在螢光幕上，這一位元組的上面那一塊的顏色，同時在高點的尼柏上的值則決定在螢光幕上低點塊的顏色。

色彩由 0 到 15 編號，如下表：

Table 8: Low-Resolution Graphics Colors

Decimal	Hex	Color	Decimal	Hex	Color
0	$0	Black	8	$8	Brown
1	$1	Magenta	9	$9	Orange
2	$2	Dark Blue	10	$A	Grey 2
3	$3	Purple	11	$B	Pink
4	$4	Dark Green	12	$C	Light Green
5	$5	Grey 1	13	$D	Yellow
6	$6	Medium Blue	14	$E	Aquamarine
7	$7	Light Blue	15	$F	White

表　　八

因此，如果一個位元組含有十進位值 $D8，則在螢光幕上將出現一棕色塊在黃色塊上面，使用十進位運算，低一點的色塊是由位元組的值除以16所得的商來決定，高一點的色塊是由所得到的餘數來決定。圖二是 Apple 的低解析度繪圖顯示形式圖及在螢光幕上的每一顯示塊的記憶體位置的位址。

由於低解析度繪圖螢光幕與主要編輯螢光幕顯示同一記憶體的區

域，如果你在編輯形式及低解析度形式之間交換做的話，就會有很多有趣的事情發生。例如：如果螢光幕在低解析度的繪圖形式，同時是充滿了彩色的色塊，同時這個編輯／繪圖選擇開關是定在編輯的形式，影幕將充滿看起來很亂的編輯字元，有時會閃動有時會轉色。相同的，如在螢光幕充滿編輯字元時觀察在低解析度繪圖形式，則會出現像長水平的灰、粉紅、綠或是黃色的桿子被很亂的色塊所分割。

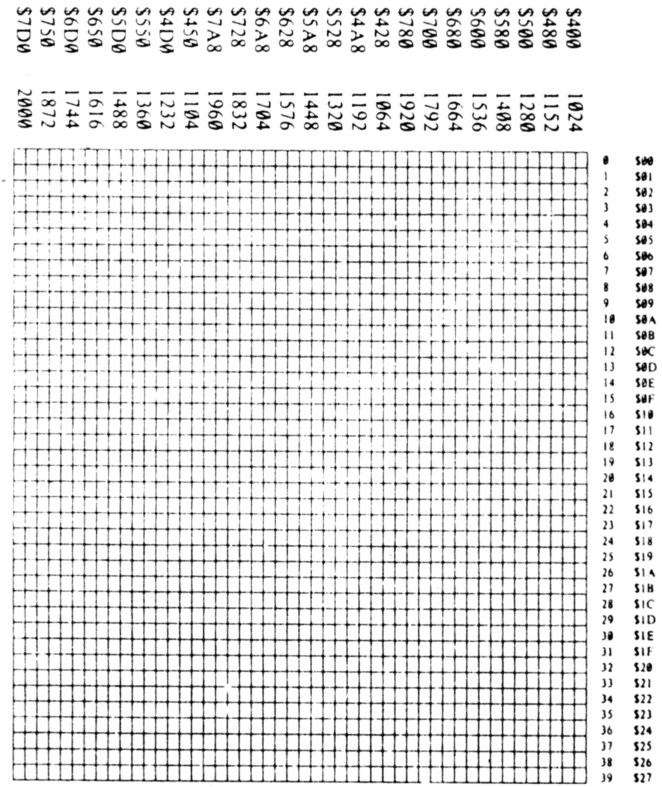


圖 二

十五、高解析度繪圖形式（THE HIGH-RESOLUTION GRAPHICS MODE）

Apple 有第二種形態的繪圖顯示，被稱爲高解析度繪圖。當你的 Apple 是在高解析度的繪圖形式時，它能顯示共 53760 個點，在一個高 192 點，寬 280 個點的矩陣內，影幕則能顯示黑、白、紫、綠、紅及藍色的點，雖然如此但對每個各別點相關的顏色仍然是有些限制的。

高解析度繪圖形式，從一個 8,192位元組的記憶區域取得資料，這一區域通常稱爲一個“畫圖暫存區”，在這裡有二個分開的畫圖暫存區，一個是爲主要頁而設，另一個則是爲次要頁而設，這二個暫存區都是獨立的，同時從記憶區分開給編輯和低解析繪圖形式使用。高解析度繪圖形式的主要頁暫存區域，由記憶位置 8192 開始，同時擴大到位置 16383。至於次要頁畫圖暫存區則接著 16383 的下一位置 16384，延伸到位置 24575。如以十六進位表示則主要頁便由$2000 到 $3FFF，同時次要頁則接著下來從 $4000 到 $5FFF。如果你的 Apple 是有 16K 或更少的記憶體，這樣次要頁對你而言是無法處理的，如果記憶體大小小於 16K 那麼整個高解析度繪圖形式對你而言是無法獲得的。

在螢光幕上的每一點卽表示在繪圖暫存區上的一位元，在每一位元組的八個位元中的七個位元被顯示在螢光幕上，剩下的位元則用來選擇這一位元上每一點的色彩。在螢光幕上的每一列可顯示 40 個位元組，在每一列上的第一位元組的第一個位元顯示在螢光幕上的最左邊，接著是第二位元，然後是第三位元等等，但第八個位元則不會顯示出來，然後就是下一位元組的第一個位元。螢幕上的192列的每一列共有 280 個點顯示在每一列上。

在一個黑白的電視機組上，每一位元相對的點如是“開”的值（或等於1）則顯現白點，若這組元相對的點是“關”值（或等於0）則顯現黑點。在彩色的顯像器上或電視機上，那就沒有這麼簡單了，如果一個位元是“關”的值，它相對的點將永遠是黑點。如果這一位元是“開”的值，那它的色彩將依照這一點在螢光幕上所在的位置而定，如果這一點是在影幕上的最左邊的一行即“第0列”或是在任何偶數列上，則這一點將顯現紫色；如果這一點是在最右邊的列，即第279列或任何奇數列，那這一點將出現綠色；如果這兩點是邊接邊的列著，那這二點都會顯現出白色；如果每一位元組上的不顯示的位置是被定爲“開”值，那藍色就取代紫色而紅色就取代綠色，因此在高解析度的繪圖形式上可以有6種顏色，主要係依照下列的規則：

㈠在偶數列上的點必須是黑色、紫色或藍色。

㈡在奇數列上的點必須是黑色、綠色或紅色。

㈢每一位元組必須是一個紫／綠位元組或是藍／紅位元組。若想把綠和藍混合，或綠和紅色混合，或紫和藍，或紫和紅混合在同一位元組上，那是不可能的。

㈣兩個邊接邊的彩色點永遠是白色，甚至於這二點是在不同的位元組上。

㈤在歐洲化的Apple上，這些規則仍可用，但是在高解析度的繪圖形式上的色彩產生器可能不同。

圖三：顯示Apple的高解析度繪圖形式螢光幕顯示，及在螢光幕上每一列的記憶體位址。

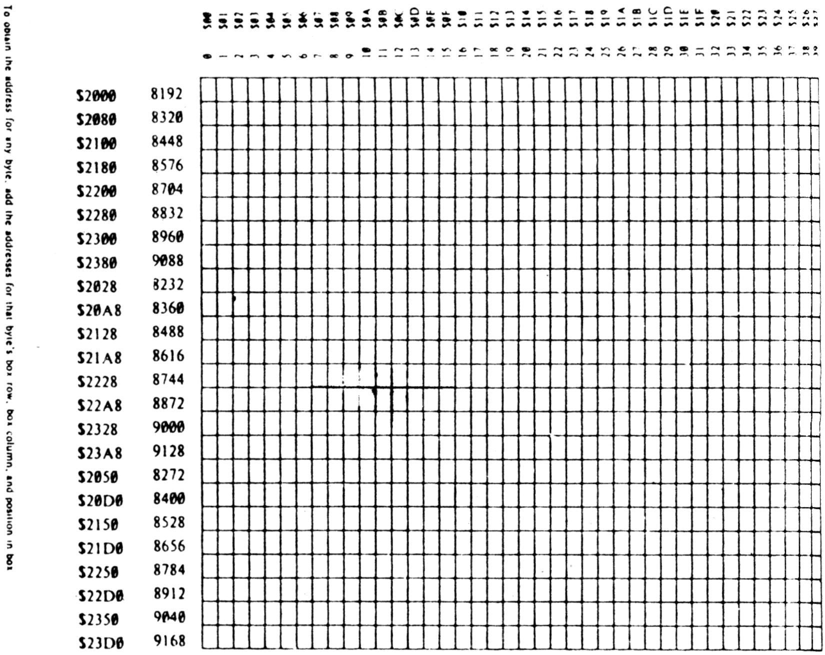
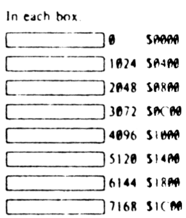


圖　　三



十六、其他的輸入／輸出情況

Apple Input/Output Features

Inputs:	Cassette Input Three One-bit Digital Inputs Four Analog Inputs
Outputs:	Cassette Output Built-In Speaker Four "Annunciator" Outputs Utility Strobe Output

㈠發音器（THE SPEAKER）

在Apple 的盒子裡面，在鍵盤下的左邊有一個小的 8 歐姆的發音器，它被連接到Apple 內部的電路，因此一個程式能夠使它發出不同的聲音。

這一發音器由一個軟體開關所控制，這一開關能夠使發音器的紙錐體出現"進"或"出"這兩個情況。這一軟體開關不像別的軟體開關可控制不同的螢光幕形式，但是可取代一個套環式開關，每次一個程式參用到與發音器開關有關的記憶體位址時，發音將改變情況，從"進"改爲"出"或從"出"改爲"進"。每一次情況改變時，發音器便產生一個很小的"咔"聲，經由連續不斷的參用發音器開關的位址，一個程式便能從發音器產生一個不停的聲響。

發音器的軟體開關與記憶位置49200相連，任何參用到這一位址，將會引起發音器發出"咔"聲。

一個程式可利用執行一個"讀"或"寫"操作到發音器的位址，就可以參用發音器的特殊位置的位址，讀或取資料兩者之間並無相關，把它當做是一位址，轉換這開關。注意：一個"寫"的操作在Apple的6502 微處理器上，事實上是先執行一個"讀"在這"寫"之前，因此如果你使用一個"寫"操作去觸動任何軟體開關，事實上你將轉

換這開關二次。如果是套環式的軟體開關，例如發音器開關，這意思就是一個“寫”的操作到這一特殊位置時控制開關將使這開關先離開它以前所在的情況。

㈡卡式磁帶介面板（THE CASSETTE INTERFACE）

在 Apple 的主要電路板背後邊緣之右邊，在螢光幕連接器的下一個位置上有二個小的黑色接點，標示“IN”及“OUT”，這裡有縮小的耳機插孔，這一插孔你可以插入兩頭有縮小的耳機插孔的電纜。電纜的另一端可連接到一個標準的卡式磁帶錄音機，因此你的 Apple 能儲存訊息到可聽的卡式磁帶同時可再讀回來。

在連接上有符號“OUT”的接點，被連綫產生另一個軟體開關。這是另一個套環開關，有如發音器的開關。卡式磁帶輸出的軟體開關塞子能因參用記憶位置數 49184 而被套住，參用這一位置將使在OUT連接器上的電壓由 0 升到 0.025 伏特，或是由 0.025 伏特回到零。如果在電纜的另一端插進卡式磁帶錄音機的麥克風輸入位置，這樣將產生一種很小的“咔”聲。錄資料到磁帶上去，使用連續的參用與卡式帶輸出有相連的記憶體位置，這樣一個程式能產生一個聲音到錄音帶上，利用改變音節及聲音持續的時間，訊息可被在磁帶上編碼，且儲存起來作爲以後使用，這個把在磁帶上的資料編碼的程式是包含在系統監督器上，在第三章中將詳述之。

預先提醒你，如果你正因觸動磁帶輸出的軟體開關，而使用寫到它的特殊位置的方式，你將實際上產生了二次的記錄，這種原因在上節發音器部份已解說過了，當你要套住磁帶輸出軟體開關時，你只要使用“讀”操作就可以了。

另一個連接器符號“IN” 的接點，能被用來聽錄音帶上的記錄，他主要的目地是提供一個方法聽在磁帶上的聲音，將它們解碼使成爲資料，同時把它們存儲到記憶體。因此，一個程式或資料組，如是

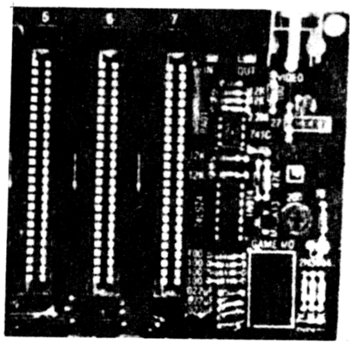
被存在磁帶上可能被讀取出來而再用一次。

輸入的綫路從卡式錄音機的耳機挿座上獲得一伏特的信號，同時把它轉換成爲一串的 0 及 1 。每一次信號供應到輸入綫路時則由正到負，或由負到正，這輸入綫路改變情況：如果它是上升，則它將由零開始上升，反之亦同。一個程式能利用尋找記憶體位置49248或相當十六進位 $C060 位置，來檢查這一磁帶輸入綫路的情況，如果從這一位置讀取到的值是大於或等於 128，那這一情況是“1”。這一值如果小於 128，那這一情況是“0”。雖然BASIC 程式能讀取這一磁帶輸入綫路的情況，BASIC程式的速度通常太過於慢，以致於對其所讀取的值無能力做任何有意義的事情。然而，這裡有一個程式在系統監督器上，可以讀取在磁帶上的聲音，同時把它們解碼，這一程式在第三章將詳述之。

㈢遊戲輸入／輸出連接器（ THE GAME I／O CONNECTOR ）

遊戲輸入／輸出連接器的目的是爲了方便我們連接特殊的輸入及輸出設備來增加程式的效能，尤其是在玩遊戲的程式（game program）時，這個連接器上可以有三個單一位元（ one-bit ） 輸入端及四個單一位元的輸出端，與一個資料特殊指示信號（ stroble ）及四個類比（ analog ）輸入端來和Apple連接，而這些均能由程式來加以控制

Apple 附有二個遊戲控制器（ Game Controller ）及電纜，我們可以將它插入此Game I／O Connector 。此控制器上的轉盤係接到連接器的類比輸入端上，至於控制器上的按鈕則接到連接器的單一位元輸入端上。



照片七　遊戲輸入／輸出連接器

①指示器的輸出（ANNUNCIATOR　OUTPUTS）

上述的四個單一位元輸出端稱爲指示器“annunciator”，此指示器可輸出訊息以做爲其他電子設備的輸入，也可以將此輸出訊息連接到磁碟機的指示燈及繼電器（ relays ）和喇叭等。每個指示器均由一個軟體開關（ soft switch ）來控制，此指示器軟體開關的位址共分成四對，每對各控制一個指示器，如果我們指示第一個位址，則將使得與其相對的指示器處於“off”狀態，如果指示的是第二個位址，則將使得指示器處於“on”狀態，當指示器處於“off”狀態時，它在連接器上的對應釘脚之電壓將接近0V，而當指示器處於“on”狀態時，則接近5V。指示器的軟體開關如下所示：

Table 9: Annunciator Special Locations

Ann.	State	Address: Decimal		Hex
0	off	49240	-16296	$C058
	on	49241	-16295	$C059
1	off	49242	-16294	$C05A
	on	49243	-16293	$C05B
2	off	49244	-16292	$C05C
	on	49245	-16291	$C05D
3	off	49246	-16290	$C05E
	on	49247	-16289	$C05F

②單一位元的輸入（ONE-BIT INPUTS）

上述三個單一位元輸入端能各自連接其他電子設備或遊戲控制器上的按鈕，我們也可以與前面所討論到磁帶輸入的方式一樣地經由機器語言或BASIC程式來讀取任一單一位元輸入端的狀態。這三個單一位元輸入端的位址係自 49249 到 49251（或－16287 到－16285，或十六進位制的 $C061 到 $C063）。

③類比輸入（ANALOG INPUTS）

上述四個類比輸入端能夠連接 150K歐姆（ohm） 可變電阻（variable resistore）或電位計（potentiometers），在可變電阻的輸入端及＋5V電源端係 one-shot時序電流，由於電阻的輸入變化因而時序電流之指標數隨之變化，機器語言程式可了解時序迴路（timing loops）的變化，並且可以得知電位計的值。

在程式開始讀入電位計值之前必須先重定時序電流，編號 49264（或－16272，或十六進位制的 $C070）的位置即從事這項工作。當我們重定時序電流時，在位置 49252 到 49255（－16284 到－16281 或十六進位制的 $C064 到 $C067）內之值將大於 128，並且在 3.060 milliseconds（千分之一秒）內此值將降到 128 以下，然而此值下降所需時間全視設定該位址的遊戲操縱桿（game paddle）而定。如果電位計連接的類比輸入端之電阻高於 150K歐姆或是根本不接電位計時，此遊戲控制器位置內之值可能永遠不會降到 0。

④特殊指示信號輸出（STROBE OUTPUT）

另一種輸出稱為C040 STROBE，通常維持在＋5V；但當機器語言或BASIC程式持續 0.5 微秒（microsecond）的控制訊號時，則會降到 0V。我們可以經由指示到位址 49216（－16320，或十六進位制的 $C04F）來啟動此特殊指示信號，如果我們執行一個儲存的動作到此位址時，會啟動此特殊指示信號兩次。

Table 10: Input/Output Special Locations				
Function	Address Decimal		Hex	Read/Write
Speaker	49200	-16336	$C030	R
Cassette Out	49184	-16352	$C020	R
Cassette In	49256	-16288	$C060	R
Annunciators*	49240 through 49247	-16296 through -16289	$C058 through $C05F	R/W
Flag inputs	49249	-16287	$C061	R
	49250	-16286	$C062	R
	49251	-16285	$C063	R
Analog Inputs	49252	-16284	$C064	R
	49253	-16283	$C065	
	49254	-16282	$C066	
	49255	-16281	$C067	
Analog Clear	49264	-16272	$C070	R/W
Utility Strobe	49216	-16320	$C040	R

表　　十

十七、APPLE的一些差異

在基本的Apple Ⅱ微電腦上有少數不同的差異，其中的一些差異在於機器本身上的修改，另外則由於其中操作軟體的改變，下面要討論的這些均是基本上的差異。

㈠自動啟動系統與監督系統之唯讀記憶體（AUTOSTART ROM MONITOR ROM）

所有的Apple Ⅱ PLUS系統均包括有Autostart及Monitor 的ROM，至於其他的Apple系統則不包含Autostart ROM ，但卻代之以Apple系統監督唯讀記憶體（Apple System Monitor ROM）。這一部份的ROM中有些特點是Autostart ROM所沒有的，但Autostart ROM中也有些部份的特點是System Monitor ROM 所沒有的，它們的主要差異為：

①編輯控制（ Editing Control ）

按 ESC 鍵後再依序按 I ， J ， K ， M 鍵，可以各使 cursor 向上、左、右、下移動，只有 Old Monitor ROM 沒有這種功能。

②停止列印（ Stop-List ）

按 CTRL S 鍵可以使得大多數的 BASIC、機器語言程式或列印的執行產生暫停，像這種 Stop-List 的特點在 Old Monitor ROM 上是沒有的。

③重定初始狀態的循迴步驟（ The Reset Cycle ）

在我們首次開啟 Apple 及按 RESET 鍵時 Old Monitor ROM 會直接使我們進入 Apple System Monitor 。（參考第二章）

Old Monitor ROM 的 System Monitor 提供了 STEP 及 TRACE 除錯（ debugging ）的特點，但是 Autostart ROM 則無法識別這兩個 Monitor 命令。

㈡Revision 0 Revision 1 電路板

一些 Revision 0 的 Apple Ⅱ電路板上所缺乏的特點可以在 Revision 1 的 Apple 電路板上找到，若要判別我們所擁有的畢竟是那一類的電路板，我們可以打開蓋子來看它的右上方角落並與照片四比較。如果在四釘脚的 Video 整流器間沒有 single metal video 連接器，那麼您所擁有的即爲 Revision 0 的 Apple。今將這兩種的不同點列述如下：

① Color Killer

當 Apple 的螢光幕顯示是 Text 模式時，在 Revision 0 的 Apple 電路板上的螢光幕輸出電路上的彩色信號將會脫離，這會出現染上色彩或鑲了色彩的 text 字元。

② Power-on RESET

當我們啟動 Revision 0 的 Apple 電路板時沒有電路來自動重新設

定一個RESET迴路，我們必須先按 RESET 鍵來啟用Apple 。此外，當我們啟動Revision 0的Apple電路板時，鍵盤即可開始作用並且假設已經鍵入一個任意的字元，而當Apple開始找尋輸出資料時即會接受此一任意的字元。爲了要清除這個字元，我們必須在開啟Apple時先按 RESET 鍵然後再按 CTRL X 鍵。

③ Colore in High-Resolution Graphics

Revision 0的Apple電路板上只能產生高解析度的彩色繪圖模式：黑色、白色、紫色、綠色。在高解析度繪圖顯示時，其每一位元組的最高位元是無效的。

④ 24K Memory Map problem

Revision 0的Apple Ⅱ電路板上擁有20K或24K RAM記憶體，當使用在BASIC中時，其記憶體比實際上要來得大，關於此部份問題請詳閱第四章。

⑤ 50HZ Apples

Revision 0的Apple Ⅱ電路板上無法經由割掉pads及 jumpers而轉換螢光幕信號成PAL或SECAM標準彩色螢光幕信號以適合歐洲電視接收器的使用。

⑥ Speaker and Cassette Interface

Revision 0的Apple電路板上經由發音器產生的聲音，均可在磁帶機介面板的OUT 連接器上作爲一個信號。如果我們離開磁帶錄音機的RECORD 模式，那麼經由發音器產生的聲音亦可以經由磁帶來予以錄製。

⑦ Cassette Input

由於磁帶介面板的輸入電路已經加以修改，因此可以很精確地接受微弱的信號。

㈢改變電源供應器（POWER SUPPLY CHANGES）

在有些Apple上只能接受110V的電源供應器，因爲在它們的電源供應器後面並沒有裝上電壓選擇開關（voltage selector switch）。

四THE APPLE Ⅱ PLUS

Apple Ⅱ Plus是擁有Revision 1電路板的標準Apple Ⅱ 微電腦，它擁有Autostart Monitor ROM及代替Apple Integer BASIC的Applesoft Ⅱ BASIC語言。歐洲型的Apple Ⅱ Plus則裝設有一個110V或220V的電源供應器；另外Apple Mini-Assembler, Floating－Point Package及SWEET－16讀卡機（interpreter）均儲存在Integer BASIC ROM內，故在Apple Ⅱ Plus中並沒有包括這些。

第二章　與APPLE交談

幾乎在Apple上的所有程式及語言均需經由鍵盤來輸入一些資料與經由螢光幕來顯示結果，在Apple的唯讀記憶體（ ROM memory ）中均儲存有成套的副程式來處理所有程式及語言的輸入、輸出工作。這些副程式均由設計者予以命名，我們只要呼叫（ Call ）這些副程式的名稱即能很便利的來使用它們。

一、標準的輸出（ STANDARD OUTPUT ）

標準的輸出副程式爲COUT，經由此副程式COUT 能在螢光幕上以Normal 或 Inverse 型式來顯示大寫英文字母、數字及符號等，另外除了RETURN, bell character ，跳行字元（ line feed character ）及跳回字元（ backspace character ）等之外，對於其他的控制字元（ control character ）並不執行。

副程式COUT自己會維護其隱匿的"output cursor"。（這個不顯示的 cursor 與我們所看得到的cursor 具有一樣的功能，它會在螢光幕的下一個輸出位置上。）每次我們呼叫副程式COUT時均會在螢光幕出現一個字元，而不論其輸出的位置上是否有其他的字元。在輸出字元後，副程式COUT會控制其隱匿的cursor 右移一個位置以便準備繼續輸出下一個字元。如果已是最右端而無法再往右邊移動時，那麼它會到下一列的最左端位置去；又如果 cursor 已是螢光幕上的最底下一列（第24列），此時螢光幕上的全部資料會往上推移一列而在原來最底下一列的位置出現新的一列， cursor 則出現在此列的最左端位置上去。

每當我們執行副程式COUT時，若輸送一個RETURN字元給COUT則cursor會移到下一列的最左端位置去；同樣的，當cursor已是螢光幕上的最底下一列，則螢光幕上的資料全部上移一列，cursor會出現在最底下新的一列之最左端位置上。

㈠停止列印（STOP-LIST）的討論

當我們在執行副程式COUT時，輸送一個RETURN 的代碼給COUT會使得COUT迅速地自鍵盤拮取一個訊息。如果我們在鍵入資料之前先按 CTRL S 鍵，則執行會暫停到我們按入另一個鍵，稱這種暫停執行COUT的動作爲停止列印（stop-list），在沒有Autostart ROM的Apple 中則無此項功能。當我們在暫停之後按入這個鍵時，COUT會自動輸出一個RETURN代碼並恢復爲一般的輸出；按其他的鍵是無法停止stop-list動作的，除非我們按 CTRL C 鍵才會結束stop-list。在我們按 CTRL C 鍵後，COUT會將先前鍵入字元之代碼輸送回程式及語言中以便輸出，在stop-list的情況時我們也可以經由按 CTRL C 鍵來停止BASIC程式或列印的執行。

跳行字元（line feed character）能導致COUT將其隱匿cursor直接跳到下一列的最左端位置。又若此隱匿cursor已在螢光幕的最底下一列，則螢光幕上的資料會全部上移一列而隱匿cursor即出現在最底下新的一列上最左端位置。

跳回字元（backspace character）則導致COUT將其隱匿cursor跳回左邊一個位置上去。若此隱匿cursor已在螢光幕上該列的最左端位置上，則此隱匿cursor會跳回上一列的最右端位置上去。又若隱匿cursor是在螢光幕上最上面一列的最左端位置，則螢光幕並不變動只是此隱匿cursor會跳到最上面一列的最右端位置去。

當COUT輸送的是一個bell character（CTRL G）時，螢光

幕並不發生變動， cursor 也不會移動，祇是會自 speaker 傳送出一個 100 HZ 的聲音。

(二)主要編譯畫面

上節所討論輸出 cursor（ output cursor ） 的各種移動，像右移、左移等，均指的是在 40字元×24 列螢光幕上的物理量移動，我們要如何通知副程式 COUT 說我們不打算使用螢光幕上全部 960個字元的畫面，而只是想使用其中某一部份的畫面；是的，我們可以將畫面予以分割並稱此為主要編譯畫面（ text window ），經由程式及語言我們可以將上、下、左端位置及寬度儲存在記憶體的四個位置中而設定此主要編譯畫面之大小。在我們儲存好主要編譯畫面大小的位置後，副程式 COUT 會依此來規劃螢光幕的畫面，而當我們的資料在螢光幕上顯示時不會超出此畫面的範圍。就是在資料產生上下整個移動時亦不會超出範圍，COUT 會控制新畫面上的資料並保護它不使它在顯示時超出範圍。

記憶體中位址 32（十六進位制的第 $20 位址）擁有分割畫面左端範圍所在的行數，一般此位址內存放著 0 值表示將螢光幕上最左邊的第 0 行定為主要編譯畫面的左端範圍，我們可以改變此位址內之值但不能大於 39（十六進位制的 $27 ），即螢光幕的最右端第 40 行不可以做為主要編譯畫面的左端範圍。位址 33（十六進位制的第 $21 位址）則擁有主要編譯畫面的寬度（即由左到右的全部行數），通常此位址內存放著 40（十六進位制的 $28 ）表示將主要編譯畫面定為與螢光幕一樣的最大寬度 40 行，我們也可以改變此位址內之值，但不能大於此 40。若是超過 40 可能會導致 COUT 將字元搬到記憶體位置上而不是在螢光幕上顯示，如此的話可能會導致我們的程式與資料產生錯誤。

位址 34（十六進位制的第 $22 位址）擁有主要編譯畫面上端範

圈所在的列數，一般此位址內存放著0值表示將螢光幕最上端的第1列定爲主要編譯畫面的上端範圍。我們在改變此位址內之值時不能大於23（十六進位制的$17），即不能超出螢光幕上最底下一列（第24列）。又位址35（十六進位制的第$23 位址）則擁有主要編譯畫面上最下端範圍所在的列數，通常此位址內存放的值爲23（即螢光幕上最底下的第24列），同樣此位址內之值亦不能大於23。總之，當我們在改變主要編譯畫面的大小時必須知道將出現新畫面上的輸出cursor之位置。

Table 11: Text Window Special Locations				
Function:	Location: Decimal	Hex	Minimum/Normal/Maximum Value Decimal	Hex
Left Edge	32	$20	0/0/39	$0/$0/$17
Width	33	$21	0/40/40	$0/$28/$28
Top Edge	34	$22	0/0/24	$0/$0/$18
Bottom Edge	35	$23	0/24/24	$0/$18/$18

表 十 一

㈢黑／白畫面顯示

我們前面討論過，副程式COUT可控制畫面以Normal 或 Inverse 型式來顯示，其輸出的型式全視位址50（十六進位制的$32位址）所儲存的內容而定。若位址50內存放的值爲255 （十六進位制的$FF）則COUT會顯示Normal 型式的黑底白字，若位址50內存放的值爲63（十六進位制的$3F）則COUT會顯示Inverse型式的白底黑字畫面，又若位址50內存放的值爲127（十六進位制的$7F）則COUT會顯示出Flashing 型式的白色／黑色反覆出現的閃爍字體而此時螢光幕上的其他字元則以Inverse 型式顯示；位址50內若儲存上述3個值以外的其他數值時，則COUT將不予理會。

Table 12: Normal/Inverse Control Values		
Value: Decimal	Hex	Effect:
255	$FF	COUT will display characters in Normal mode.
63	$3F	COUT will display characters in Inverse mode.
127	$7F	COUT will display letters in Flashing mode, all other characters in Inverse mode.

表 十 二

如上所述Normal／Inverse 型式的內在位置之操作方式，仍是將位址 50 內存放之值與我們所要輸出顯示字元的代碼經由邏輯運算AND”起來；所以位址50中任一位元（ bit ）之值若是0的話，經“AND”輸出字元代碼後會使得其結果變為0，因此當位址50中之值存放63（十六進位制的＄3F或是二進位制的00111111）時，很明顯的會使得輸出字元代碼的前面兩個位元變成0，也就是說會使得輸出字元代碼變成介於0與63之間（即00000000與00111111之間），由第一章的表7中我們可以看到在0與63的範圍內之字元代碼均是以 Inverse 型式顯示。

二、標準的輸入（ STANDARD INPUT ）

標準的輸入相關副程式有：

(一)RDKEY，經由按鍵拮取輸入訊息。

(二)GETLN，經由按鍵將一連串訊息自輸入路線（ input line ）輸入。

(一)RDKEY副程式

RDKEY的基本功能是等待使用者由鍵盤鍵入訊息，然後將此訊息送到需要的程式中去。除此之外RDKEY亦執行另外兩樣工作：

①快速的輸入：

當執行RDKEY時，首先將隱匿的輸出cursor（output cursor）變成看得見的輸出 cursor ，這個動作有兩種作用：第一就是讓使用者知道 Apple 在等待鍵入一個訊息，第二就是顯示了輸入時將在螢光幕上出現的位置。在大部份的情況中，設計程式時會先輸出一段要求輸入一個訊息的字句，將要輸入的訊息緊接著該字句之後輸入，此時以 Flashing 型式出現之輸入 cursor（ input cursor ）與輸出 cursor 正好重疊。通常由於這時候 cursor 的位置上尚是空白的字元，所以輸入 cursor 大部份時間均是以閃爍（ Flashing ）的“■”顯示，等我們輸入訊息後，RDKEY 會將 cursor 移掉而經由程式將輸入的訊息取代該位置顯示出來。我們可以這麼說，輸出 cursor 只是在於指示螢光幕上的輸出位置，而輸入 cursor 則是一個螢光幕上的閃爍字元，兩者均是一起移動且甚少分開，即使輸入 cursor 消失了但是輸出 cursor 仍然繼續動作。

②用來產生亂數：

當機器在等待使用者鍵入訊息時，RDKEY 會持續地把記憶體中的一組數字加上 1 。這一組數字是由兩個記憶體位置所合併而成的（其數值範圍可以由 0 到 65,536 ），當使用者鍵入訊息後，這個數字我們是無法知道的，所以在許多的程式及語言中均使用此數字來產生亂數。這兩個記憶體位置是 78 與 79（十六進位制的第 $ 4E 與 $ 4F 位置）。

㈡GETLN 副程式

Apple 的輸入工作大部份均經由輸入路綫來完成，Apple ROM 中的GETLN 副程式即是由鍵盤取得輸入路綫，每取得一筆資料即會送往要求資料的程式。

每當我們呼叫（ Call ）GETLN時，首先會顯示一個prompting

character 或“prompt”(例如:✳,>,],!),prompt 可以表示到底是何種程式呼叫GETLN要求輸入資料。其中的prompt “✳”表示System Monitor ;prompt “>”表示Apple Integer BASIC ;prompt “]”表示Applesoft Ⅱ BASIC,prompt “!”表示Apple Mini-Assembler ;另外尚有“?”則使用於許多的程式及語言中用來表示使用者的程式要求輸入資料。以使用者的觀點來看,看到的是螢光幕上顯示一個 prompt 與輸入 cursor,而當使用者鍵入資料時會顯示出來,同時 cursor 會依次向右移動;當使用者按入 RETURN 鍵後,整列的資料會送到我們所交談的程式及語言去而 cursor 則會出現在新的一列上。

當prompt 顯示後實際上發生了那些情況:首先GETLN會呼叫RDKEY而產生輸入 cursor ;在呼叫RDKEY而鍵入資料後GETLN即將此資料儲存在輸入緩衝器(input buffer)內並且予以顯示在輸入 cursor 的位置上,緊接著繼續呼叫RDKEY重覆這個動作一直到我們按 RETURN 鍵為止;當GETLN接收到RETURN 代碼時,立即將此代碼置於 input buffer 的最後位置上,同時清除 input cursor後面的資料,再將RETURN代碼發送給COUT 副程式;最後GETLN將整列的資料交到程式去,此資料即是GETLN 儲存在 input buffer 上的資料。

當我們在一列上鍵入資料時,可以按 CTRL X 鍵來清除該列上的全部資料,GETLN會將我們鍵入的資料全部清除並且顯示“\”,最後會跳到下一新的列上去。在跳到新的一列上時,會再出現一個 prompt 而允許我們鍵入新的資料。GETLN每次在一列上可以處理資料的最大量為255個字元,如果超出這個範圍GETLN會將整列的資料清除,因此當我們鍵入的資料到達第249個字元開始,GETLN會發出聲音來警告我們,以免超出255個字元後,全部資料都被清除

而白費功夫。

當我們鍵入資料後發現按錯鍵的時候，經由GETLN我們可以直接在該列上加以修正，在此介紹Backspace（←）鍵與Retype（→）鍵以便利我們修正資料。

①The Backspace（←）Key：

每按一次←鍵均會導致GETLN將 input line 上的 cursor 向左移一個位置，此時再鍵入的資料將代替原有 input line位置上的資料並顯示出來，我們也可以重複地按←鍵而回到 input line最左端的第一個位置上去再重新鍵入資料。

②The Retype（→）Key：

每按一次→鍵均會導致GETLN將 input line 上的cursor 向右移一個位置，當我們利用←鍵到某一位置去更改資料後，可利用→鍵再回去繼續鍵入資料。若與pure cursor moves（下節會討論）配合使用，則可用來複製及編排螢光幕上的資料。

三、脫離狀況的代碼（ESCAPE CODES）

當我們按 ESC 鍵時即表示進入了脫離模式（escape mode）或編排模式（edit mode），在此脫離模式中的十一個鍵各有不同的意義，統稱之爲脫離代碼（escape codes）。當我們按其中一個鍵時，Apple 即會執行該鍵所具有的特殊功能；當執行完畢時，Apple 可以繼續脫離模式也可以停止此脫離模式，全看該鍵的執行上是否需要而定。如果我們所按的並不包括在escape code 內，則將不予理會並且停止此脫離模式。

Apple 中所承認的有十一個 escape code，其中的八個由於只是單純的移動 cursor 而不改變螢光幕或 input line上的資料，稱此八個 escape code 爲單純 cursor 移動（pure cursor moves）；另外

的三個 escape code 由於係清除螢光幕上的部份或全部資料，我們稱之爲螢光幕清除代碼（ screen clear codes ）。三個 screen clear codes 及 pure cursor moves 的前四個脫離代碼（ escape codes @，A，B，C，D，E，F ）在每次執行操作之後隨即停止脫離模式，但執行 pure cursor moves 的後四個脫離代碼（ escape codes I，K，M，J ）時，只須按 ESC 鍵一次即可重複執行，直到按其他非 escape codes 爲止。今說明此十一個 escapes codes 如下：

ESC A ：在按 ESC 鍵後繼續按A鍵，將使 cursor 向右移動一個位置，但並不改變 input line 上的任何資料。

ESC B ：在按 ESC 鍵後繼續按B鍵，將使 cursor 向左移動一個位置，但並不改變 input line 上的任何資料。

ESC C ：在按 ESC 鍵後繼續按C鍵，將使 cursor 向下移動一個位置，但並不產生左右位置的移動。當cursor 已位於螢光幕的最底下一列時，將使螢光幕上的資料全部上移一列，而 cursor 則仍在最底下一列的位置上。

ESC D ：在按 ESC 鍵後繼續按D鍵，將使 cursor 直接跳上一列去，但並不產生左右位置的移動。當 cursor 已位於螢光幕的最上面一列時，則 cursor 仍停留原位置。

ESC E ：在按 ESC 鍵後繼續按E鍵，將清除 cursor 所在列上自 cursor 位置起至最右端的全部螢光幕上資料，但並不會清除 input line 上的資料， cursor 則停留在原位置上。

ESC F ：在按 ESC 鍵後繼續按F鍵，將清除 cursor 右端及其下方各列的全部螢光幕上資料，但並不清除input

line 上的資料， cursor 則停在原位置上。

ESC @ ：在按 ESC 鍵後繼續按 SHIFT 鍵再按 @ 鍵，則整個螢光幕畫面將全部清除並且 cursor 會移到螢光幕的最左上方位置，此時 input line 上的資料保持不變。

ESC K， ESC J， ESC M， ESC I ：此四個 escape codes 的功能與上述的 ESC A， ESC B， ESC C， ESC D 一樣。祇是此四個 escape codes 可重複地執行，今圖示其 cursor 的移動方向如下：

D
I
↑
B J ←　　→ K A
↓
M
C

圖　　四

四、重定週期（THE RESET CYCLE）

開啟 Apple 或按 RESET 鍵均會使得 Apple 的 6502 微處理機開始執行重定週期（RESET Cycle）。首先是跳到 Apple 的 Monitor ROM 中的副程式去執行，在此 ROM（唯讀記憶體）中包括有 Monitor ROM 與 Autostart ROM 這兩種不同的 RESET Cycle 。

(一) Autostart ROM RESET

Autostart ROM的RESET Cycle首先控制螢光幕以Text及Low-Resolution Graphics primary page模式來顯示，隨之開啟40字元× 24列的text window並將output cursor置於螢光幕最底下一列的左端。以上均是以Normal型式的畫面來顯示，接著安排好COUT及KEYIN的開關以便使用鍵盤及螢光幕來做爲輸入／輸出的設備，並且將遊戲輸入／輸出連接器（Game I／O Connector）上編號0與編號1之指示器（annunciator）置於“ON”狀態而編號2與編號3則置於“OFF”狀態。另外還清除鍵盤上的特殊指示信號（clears the keyboard strobe）及關閉所有的I／O Expansion ROM，最後會發出“beep”聲音。

我們每按一次 RESET 鍵時均會停止其他所有的動作而重新開始執行上述的一連串動作。Autosart ROM會找出記憶體中的兩個特殊位置之內容來看，如果Apple是剛剛開啟使用的話那麼Autostart ROM會執行“Cold start”，若Apple是早已開啟並且RESET過的話那麼Autostart ROM會執行“Warm start”。今分述如下:

①Cold Start：

在剛開啟的Apple中，RESET Cycle首先清除螢光幕然後在螢光幕上方的中央顯示“APPLE Ⅱ”。隨之在記憶體中的特殊位置上表明已開啟機器與RESET過了，然後在Apple主要電路板的七個插座（slot）上由編號7開始往編號0尋找Disk Ⅱ控制面板(Controller Card)。若找到了此控制面板，即經由控制面板所連接磁碟機（disk driver）上之磁碟片（diskette）來啟動Apple磁碟作業系統（DOS），有關此啟動步驟之描述可參考Do's and Don'ts of DOS（零件編號爲A2L0012）。若無法找到此控制面板，或是在啟動DOS的動作尚未完成之前又按 RESET ，那麼RESET cycle會持續地“lukewarm start”，亦即會將Apple置於初始狀態並跳

到存放於ROM中的語言去執行。對於Revision 0 而言，不論是沒有插上Applesoft Ⅱ Firmware 面板或是有插上但控制的開關却扳到DOWN的位置，均使得Autostart ROM去執行Apple Integer BASIC語言。至於對插有Applesoft Ⅱ Firmware 面板而開關扳到UP位置或是Apple Ⅱ－Plus 而言，均使得Autostart ROM去執行Applesoft Ⅱ Floating-Point BASIC語言。

②Warm Start

若是Autostart ROM已經做過cold start 週期（ cycle ），則每當我們按 RESET 鍵後，均會使得執行回到我們所使用過的語言中去，並且會原封不動地保存著程式與變數。

㈢Autostart ROM的特殊位置

在此將討論上面提到的特殊位置，依此特殊位置可以判斷Autostart ROM是否已經做過cold start cycle 。在隨機取存記憶體（ RAM memory ）中保留了三個特殊的位置（ special location ）做爲Autostart ROM的系統功能，今圖示如下：

Table 13: Autostart ROM Special Locations

Location: Decimal	Hex	Contents:
1010 1011	$3F2 $3F3	Soft Entry Vector. These two locations contain the address of the reentry point for whatever language is in use. Normally contains $E003.
1012	$3F4	Power-Up Byte. Normally contains $45. See below.
64367 (-1169)	$FB6F	This is the beginning of a machine language subroutine which sets up the power-up location.

表 十 三

當開啟Apple時，Autostart ROM會將一個特殊的值存入上表中的power-up位置（十進位制的1012或十六進位制的$3F4位置

），這個值是由位置 1011 內之值與常數 165 經 Exclusive- OR 運算而得。例如，設若位置 1011 之值為 224 ，則與 165 經 Exclusive - OR 後得到 69 。

	Decimal	Hex	Binary
Location 1Ø11	224	$EØ	111ØØØØØ
Constant	165	$A5	1Ø1ØØ1Ø1
Power-Up Value	69	$45	Ø1ØØØ1Ø1

由於我們的程式能改變表十三中 soft entry vector 位置（十進位制的 1010 與 1011 或是十六進位制的 $ 3F 2 與 $ 3F 3 位置）的值，因此當我們按 RESET 鍵而進入程式或語言中時，要確定我們所要進入的目的地。如果我們要改變 soft entry vector 位置之值，要確定此新的 soft entry vector 中位置 1011 之值與 165 Exclusive-OR 後的結果要設為 power-up 位元組之值。若未設定此值，則下次當我們按 RESET 鍵時，Autostart ROM 會將 Apple 當做剛剛啟動而執行 cold start cycle 。

譬如，我們可以指示改變後的 soft entry vector 值到 Apple 系統監督程式（ Apple System Monitor ）去，當我們按 RESET 鍵後即會進入此系統監督程式去執行。要達到此項目的，首先要將系統監督程式的起始位址存到 soft entry vector 的兩個位置中。此起始位址為 $ FF 69（或是十進位制的 65385 ），故要將其中的 $69 存到位置 $ 3F 2 中，將 $FF 存到位置 $ 3F 3 中。若使用的是十進位制，則將 105 存到位置 1010 中，將 255 存到位置 1011 中。現在我們要設定 power-up 位置內之值，由於 Autostart ROM 中有一機器語言副程式能自動執行此項工作，所以我們只要以指令 JSR (Jump to Subroutine ）跳到此副程式中（位址 $FB6F）去執行即可。如果是以 BASIC 在做的話，則只要 CALL -1169 即可。在設定完成後，當

我們下次按 RESET 鍵時即可進入系統監督程式中去執行。若要使 RESFT 鍵回到原來的動作，只要將其原始之值（ $E003 或十進位制的 57347 ）重新存回 soft entry vector 位置，並且再次執行自動設定 power-up 值的副程式即可。

㈢“OLD MONITOR” ROM RESET

Apple Ⅱ Monitor ROM中的RESET cycle之一連串動作係先設定爲Normal Video 模式，隨之設定全部螢光幕畫面的 Primary Page Text 並混合彩色繪圖模式來顯示。接著開啟 40 字元× 24 列的 text window並設定鍵盤及螢光幕做爲輸入／輸出的設備，緊接著發出 “beep” 的聲音，並且使 cursor 跳到螢光幕的最底下一列的左端位置去。最後以 prompt “*” 出現在我們眼前，而開始使用Apple 的系統監督程式。

第三章　系統監督程式

本章所討論的主題是儲存於Apple的唯讀記憶體（ ROM ） 之內的系統監督程式（System Monitor ），它所扮演的角色既是系統監督者（ supervisor ）亦是從屬者（ slave ），它控制著所有程式的執行也可由程式來使用它。經由此系統監督程式，我們可以窺視全部記憶體內65,536個位元組（ byte ） 的內容，同時也可以：

㈠改變任何位置的內容。

㈡直接使用Apple的微處理機(microprocessor ）來執行機器語言（ Machine Language ）與組合語言（ Assembly language ）。

㈢使用磁帶（ cassette tape ）來儲存與讀取大量的資料和程式。

㈣使用簡單的命令（ command ）來搬取和比較記憶體內的上千個位元組。

當然，我們也可以不必經由監督程式而直接將任何程式及語言輸入Apple內。

一、進入系統監督程式

Apple的系統監督程式（ System Monitor)係儲存在記憶體位置＄FF69(或十進位制的65385與－ 151 ）。要進入到監督程式去熟悉其面貌如何，我們可以經由BASIC語言的Call 指令直接的呼叫出這個位置，此時在螢光幕的左端會出現Monitor的 prompt “＊”與cursor “█”，Monitor 可以像Apple 上其他的

系統或語言一樣的經由線入路線（ input line ）來輸入資料。每輸完一連串資料即按[RETURN]鍵來表示每一行上資料的終止，每一行上的一連串資料不可超過255個字元。當我們要停止執行Monitor而回到原來的程式上去時，我們只要按入 [CTRL C]鍵與 [RETURN] 鍵即可，或是乾脆按 [RESET] 鍵就可以了。

二、位址與資料的使用

同Monitor 交談跟一般在Apple上與其他程式及語言交談是一樣的，先要由鍵盤來鍵入一行敍述再按 [RETURN] 鍵，此時Monitor會將我們所鍵入的指令予以消化吸收並付諸實行。我們要讓Monitor瞭解的訊息包括有三種型式，即位址（ address ），值（ Value ）和命令（ commands ）。其中位址與值是以十六進位制來表示（即 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。其中A等於十進位制的 10 ，……，F 等於十進位制的 15 ），所以一個十六進位制的數元（ digit ）所表示之值可以表示十進位制的由 0 到 15 之值，因而四個十六進位制的數元之值可以表示十進位制的 0 到 65,536 。在Apple 內所有的位址均能以十六進位制的四個數元來表示，並且所有的值均能以十六進位制的二個數元來表示，也就是當我們與Monitor 交談時必須以此四個長度與二個長度的數元來表示其位址與值。因爲Monitor 在尋求位址時同樣地去一次取四個數元的長度，它將此四個一組的數元以十六進位制來看待，若少於四個數元時不夠的數元一律以 0 來看待，若超過四個數元時則只取後面的四個數元。同樣的，Monitor 在尋求其資料的值時亦只取二個十六進位制的數元長度。

監督程式（ Monitor ）可以辦認 22 種的命令字元，其中有些爲標點（ punctuation marks ）其他爲大寫文字（ upper-case letters

）或控制字元（ control characters ）。在接下去的各節中準備將每一種命令的整個名稱標示出來，而Monitor所需要的只是每一種命令名稱的第一個字，其中有些命令是用來取得控制字元的。必須注意的是，即使Monitor能予以辨認與解析這些命令字元，但却不一定會在螢光幕上顯示出來。

Monitor能夠記得的位址有五個位置，其中有兩個比較特殊的位址爲：

㈠ last opened location，此位址是使用者所要的所有位址中最後的一個，Monitor 中記憶著此一位址那麼在執行時才能知道是要執行到何時即需停止。

㈡ next changeable location，此位址是Monitor 所要執行的下一個位址，有了這一個位址的"指示"Monitor 才能正確的執行下去。

三、查詢記憶體的內容（ EXAMINING THE CONTENTS OF MEMORY ）

㈠當我們想要查詢記憶體內某一個位置的內容時，我們只要將該位置的位址予以輸入即可，此時我們可以看到我們所輸入的位址之後緊接著一個 dash ㈠及一個空白 ，然後便是該位址所儲存的值。例如

```
*E000

E000- 4C
*300

0300- FF
```

（說明）① E 000 即爲我們所鍵入的要查詢的位址， 4 C即爲其值。

② 3 0 0　爲我們所鍵入的位址，FF即爲位址 300 內之值。

每次當Monitor顯示某一位址所含的值時，Monitor即會將該位址視爲 last opened location，如上例中在執行E000 時Monitor即視E000爲 last opened location ，在此時Monitor 同樣將該位址視爲next changeable location 。

(二)在上面所討論的是我們每次只查詢記憶體中一個位址的內容，如果我們想要一次查詢一連串的資料時應該怎麼做呢？在此，我們可以看到有兩種方式可以達到此項目的：

(A)在"✳"之後先鍵入一個句號"。"然後再鍵入我們所要查詢資料的位址即可。此時Monitor會將一連串的資料顯示出來，稱此種情形爲記憶體傾印（memory dump）。此時Monitor 會自 last opened location 開始到我們所鍵入位址之間所有位址的內容顯示出來，此時的 last opened location及next changable location 均爲螢光幕上所顯示的最後一個位址。例如：

```
*20

0020- 00
*.2B

0021- 28 00 18 0F 17 A2 FF
0028- D0 07 D0 07
*300

0300- 01
*.315

0301- AD 00 E0 48 AD 81 C0
0308- 68 48 CD 00 E0 D0 23 AD
0310- 83 C0 AD 83 C0 A9
*.32A

0316- A5 8D
0318- 00 D0 CD 00 D0 D0 13 4A
0320- 8D 00 D0 CD 00 D0 D0 0A
0328- AD 81 C0
```

（說明）①只顯示位址 20 之內容 00，此時之 last opened location 及 next changeable location 均爲位址 20。

②當我們鍵入“．2B”做第二次查詢時，由於 next change -able location 爲位址 20，而此時之 last opened loca-tion即爲位址 2B，故自位址 21 到位址 2B 之內容將全部 dump。

③當我們做第三次查詢而鍵入“300”時，由於只查詢位址 300，故只顯示位址 300 之內容 01。此時之last opened location 與next changeable location 均爲位址 300。

④當我們做第四次查詢而鍵入“.315”時，由於next chang - eable location爲位址 300 故自位址 301 到位址 315 的內容均會全部dump。在顯示後，last opened location及 next changeable location 均爲位址 315。

⑤當我們做第五次查詢而鍵入“.32A”時，即會將位址316 到位址 32A的內容全部dump 出來。

我們可以看到，dump 顯示出來的格式是以 0 與 8 做爲顯示時每段的開頭，而且每段上的資料均不超出 8 個。也就是說，我們 dump 出來的資料在遇到最右邊一位有 0 或 8 的位址時即會在新一列上顯示。另外，當我們在做 dump（即鍵入位址之前先鍵入一個句點“.”）時，由於Monitor 均是以 last opened location 爲依據，若我鍵入的位址小於 last opened location 時，Monitor將不會顯示我們所要的位址及內容，而只顯示 last opened location的後面一個位址之內容。例如：

```
*.315

0301- B9 00 08 0A 0A 0A 99
0308- 00 08 C8 D0 F4 A6 2B A9
0310- 09 85 27 AD CC 03
*
 FF FF
*.315

0318- FF
*.308

0319- FF
```

（說明）由於 dump完位址0301到0315後， last opened location 及 next changeable location 均為位址 0315 ，故當我們再要 dump 時，Monitor 不管鍵入的位址 0315 而只顯示位址 0318 。至於例中所出現的 "FF FF"是因為在顯示位址 0301 到位址 0315 後，我們按了 RETURN 鍵而會繼續顯示位址 0310 之列上未完的兩個位址（即位址 0316 與位址 0317 的內容為FF與FF），此時之 last opened location 為位址 0317 ，所以造成我們想繼續 dump 時出現了位址 0318 的內容。

在這種情況時，若我們想繼續 dump 的位址大於 last next location 時則毫無問題，例如：

```
*.32A

031A- FF FF FF FF FF FF
0320- FF FF FF FF FF FF FF FF
0328- FF FF FF
```

（說明）由上例中，我們知道最後的 last opened location 爲位址 0319，故當我們繼續鍵入位址 032A 來 dump 時，螢光幕上會顯示自位址 031A 到位址 032A 的內容。

但是當我們想回到位址較低的位置去查詢資料時應該怎麼辦呢？首先我們必須只先查詢一個單獨的位址來使得 last opened location 回到較小的位址上去，然後再 dump 。例如：

```
*300

0300-  99
*.315

0301-  B9  00  08  0A  0A  0A  99
0308-  00  08  C8  D0  F4  A6  2B  A9
0310-  09  85  27  AD  CC  03
*.32A

0316-  85  41
0318-  84  40  8A  4A  4A  4A  4A  09
0320-  C0  85  3F  A9  5D  85  3E  20
0328-  43  03  20
```

（說明）我們可以很明顯的看出，由於位址 0300 的查詢使得Monitor 的 last opened location 又回到較小的位址去。此時，我們可以繼續再 dump 較小位址的內容了。

(B)我們可以將要 dump 的起始位址與最終位址鍵入其間以句點“ ”加以隔開，則可以依我們的需要來 dump 一個範圍內的全部位址。例如：

```
*300.32F

0300- 01 AD 00 E0 48 AD 81 C0
0308- 68 48 CD 00 E0 D0 23 AD
0310- 83 C0 AD 83 C0 A9 A5 8D
0318- 00 D0 CD 00 D0 D0 13 4A
0320- 8D 00 D0 CD 00 D0 D0 0A
0328- AD 81 C0 AD 81 C0 A9 01
*30.40

0030- FF 00 FF AA 05 00 BD 9E
0038- 81 9E 49 03 3C 00 40 00
0040- 30
*E015.E025

E015- 90 FB 20
E018- 7D E0 B0 F6 C9 24 D0 06
E020- A9 FF 85 11 D0 10
```

（說明）在我們鍵入起始位址的 300 與最終位址的 32F 後，在這個範圍之內的位址全部被顯示在螢光幕上。我們可以不必在在高位址去查詢後，再使用鍵入單一位址的方式來回到低位址去查詢。如例中所示，我們可以直接的由位址 32F回到位址 30去。

㈢前面提到的例子中我們曾經由按入 RETURN 鍵來 dump未完的部份顯示段上的資料，在本節中我們將加以說明。要知道，每按一次 RETURN 鍵最多只能 dump 一列（即 8 個位址）的內容，同樣的以 0 與 8 做爲固定分段的位置。例如：

```
*5

0005- DB
*
 0C 00
*

0008- 00 00 4C 99 E1 00 00 6B
*32

0032- FF
*
 AA 00 00 BD 9E
*

0038- 81 9E 49 03 3C 00 3F 00
```

（說明）①第一次只查詢位址0005，此時其last opened location與next changeable location均為位址0005。

②當我們按 RETURN 鍵做第二次查詢時，由於此時之last opened location 為位址0005，故Monitor 會將該段所剩餘的位址0006與位址0007之內容顯示出來。

③當我們再按 RETURN 鍵時，Monitor會將位址0008開始的一段位址之內容全部顯示出來。

四、改變記憶體位置的內容

在本節中所要討論的是如何去改變記憶中某個位置上的內容。我們可以先查詢該位址之內容，然後在鍵入該位址的新內容之前先鍵入一個冒號“：”即可。我們也可以直接鍵入位址後，再鍵入一個冒號與新的內容。例如：

```
*0

0000-  4C
*:5F

*0

0000-  5F
*302:42

*302

0302-  42
```

（說明）①先查詢位址 0000 的內容，看看是否確定要改，此時之 last opened location 爲位址 0000。

②由於Monitor 在改變內容時同樣以 last opened location 爲依據，故此時在冒號後鍵入 5F 將使得位址 0000 之內容變爲 5F。

③當我再去查詢位址 0000 時，很好！它已經正如我們所需要的變爲 5F。

④我們直接在位址後面更改其內容，同樣的有效。只是在這種情形下，我們無法知道位址 0302 原來的內容究竟如何了。除非我們已確知其內容並且決定要予以更改，否則一經更改，其原來的內容將不復存在而已被新的內容所取代了。

五、更改一連串位址的內容

如果我們想更改一連串位址上的內容時，我們只要鍵入該串位址的第一個位址並在鍵入一個冒號之後即可開始鍵入一連串新的內容。

在每兩個內容之間必須以一個空白加以隔開，Monitor 允許我們每次最多可更改 85 個位址的內容。若一列放不下時，可接下一列繼續，只是要在下一列先鍵入一個冒號後再鍵入新的內容。例如：

```
*300:69 01 20 ED FD 4C 0 3

*300

0300- 69
*
 01 20 ED FD 4C 00 03
*10:0 1 2 3

*:4 5 6 7

*10.17

0010- 00 01 02 03 04 05 06 07
```

（說明）①自位址 0300 開始，我們連續鍵入 8 個新的內容。

②我們首先查詢位址 0300 的內容是否已經更改，此時之 last opened location 即爲位址 0300。

③當我們按 RETURN 鍵來繼續查詢該段未完的位址時，螢光幕上顯示出位址 0301 到位址 0307 的內容。

④我們繼續鍵入位址 0010 開始的 4 個位址之新的內容。

六、記憶體區域的搬移

接著我們所要討論的是將某一個範圍之內的資料搬移到其他位置上去，這一連串的動作稱爲Monitor 的MOVE命令。要達到這一連串的動作，首先要使Monitor 瞭解是搬運那一塊區域與搬到那個位置去。當然我們要告訴Monitor 我們所要搬移區域的起始位址及最終位址，同時也要將準備放置這些內容的目的位址告訴Monitor，另外也得讓Monitor 知道我們所要執行的是一個搬移的命令。其格式如下：

{目的位址} < {起始位址} . {最終位址} M

其中．(一)“<”是用來隔開所要搬移的原始區域與放置這個原始區域的目的區域。

(二)“.”是用來隔開所要搬移區域的起始位置與其最終位置。

(三)“M”是用來表示欲執行的是“MOVE”命令。

(四)目的位址係放置此搬移區域的起始位置，由此位置開始將該搬移區域的內容依序放置於其後面的位置上。

（例）

```
*0.F

0000- 5F D4 D4 4C 3A DB FF FF
0008- FF FF 4C 99 E1 22 00 6B
*300:A9 8D 20 ED FD A9 45 20 DA FD 4C 00 03

*300.30C

0300- A9 8D 20 ED FD A9 45 20
0308- DA FD 4C 00 03
*0<300.30CM

*0.C

0000- A9 8D 8D ED FD A9 45 20
0008- DA FD 4C 00 03
*310<8.AM

*310.312

0310- DA FD 4C
*2<7.9M

*0.C

0000- A9 8D 8D DA FD A9 45 20
0008- DA FD 4C 00 03
```

（說明）①鍵入“0.F”，則會將位址0000至位址000F之內容dump出來。

②鍵入“300:”，表示自位址0300開始更改內容。

③鍵入“300.30C”，表示dump位址0300到位址030C。主要目的在檢查我們所更改的內容是否已經儲存在記憶體內。

④鍵入“0<300.30CM”，表示將位址0300到位址030C的內容搬移到位址0000開始到位址000C的位置去。

⑤鍵入“0.C”，表示dump位址0000到位址000C的內容出來看是否已放置由位址0300到位址030C的內容了。

⑥鍵入“310<8.AM”，表示將位址0008到位址000A爲止的3個位址之內容搬移到位址0310開始的3個位址去。

⑦鍵入“310.312”，表示dump位址0310到位址0312爲止的3個位址之內容出來看，是否已由位址0008到位址000A搬移過來了。

Monitor的MOVE命令可以很方便的將記憶體中某些位置上的資料搬移到其他位置上去，MOVE命令執行之後，原有位址上的內容並不改變，只是被copy到其他位置上去而取代了其他位置上的原有資料。應該注意的是：

(一)如果最終位址小於起始位址，則只有起始位址的內容被搬移到其他位置去。因爲Monitor所記憶著的last opened location即爲最終位址，而next changeable location即爲起始位址，所以一開始執行時即將起始位址之內容MOVE到目的位址上去，但當Monitor要MOVE下一個位址的內容時，由於next changeable location大於last opened location故不再執行MOVE命令。

㈡如果目的位址介於起始位址與最終位址之間，則由起始位址到目的位址間的原有內容不變，但目的位址到最終位址之間的內容則發生有趣的改變，我們稍後會討論這種特殊的變化。

七、兩個記憶體區域的比較

Monitor 能夠比較兩個記憶體區域的資料是否相同，稱這種執行比較的命令爲VARIFY。一般做比較的VARIFY命令均緊接在MOVE命令之後來使用，其目的爲檢查MOVE命令之執行是否正確。VARIFY命令之格式如下：

｛目的位址｝ < ｛起始位址｝ . ｛最終位址｝ V

Monitor 在執行VERIFY命令時，係自起始位址到最終位址爲止的範圍內之資料依序與目的位址開始的資料進行逐一的比較。若比較的結果完全相同則不會出現任何訊號，若有不同則會顯示出此不同的位址及其內容。

（例）

```
*0:D7 F2 E9 F4 F4 E5 EE A0 E2 F9 A0 C3 C4 C5

*300<0.DM
*300<0.DV

*0.D

0000- D7 F2 F2 F4 F4 E5 EE A0
0008- E2 F9 A0 C3 C4 C5
*6:E4

*300<0.DV

0006-E4 (EE)
```

（說明）①首先更改由位址0000開始的14 個位址之內容。

②鍵入“300<0.DM”，表示將位址0000到位址000D之內容MOVE到位址0300開始的14 個位址上去。

③鍵入“300<0.DV”，表示比較上述兩個區域之內容是否有不同的，以便確定MOVE命令的執行是否正確。

④鍵入“6:E4”，表示更改位址0006之內容。

⑤鍵入“300<0.DV”，重新再比較兩個區域之內容，結果出現了不同內容的位址爲0006（其內容爲“E4 ”）與其相對應的位址0306（其內容爲“EE”）。

VERIFY 命令的執行並不會改變原始區域與目的區域之內容，另外與MOVE命令相同的是如果標示原始區域的最終位址小於起始位址時，VERIFY命令只執行此兩個區域中第一個位址上的內容之比較後便停止執行。又如果標示目的區域的目的位址介於原始區域的起始位址與最終位址之間時，則與MOVE命令一樣的發生有趣的變化，我們稍後再討論。

八、存放記憶體的資料到磁帶機上

Monitor 包含有兩個特殊的命令，經由此兩個特殊的命令，我們可以將記憶體中的部份資料儲存到磁帶上去，也可以隨時再由磁帶上搬回記憶體中。我們所要介紹的第一個命令稱爲WRITE ，這個命令可以使我們將一個到65,536 個記憶體位置上的資料存放到磁帶上去。而欲達到此項目的，我們必須告訴Monitor，我們要存放到磁帶去的資料在記憶體中的起始位址與最終位址，另須加上“W”這個字來表示WRITE 命令的執行。其格式如下：

{起始位址}　　{最終位址} W

我們在按 [RETURN] 鍵執行此WRITE 命令之前，要記得先將磁帶機開啟並開始做 record 工作，先讓磁帶轉動一段時間後我們再去按 [RETURN] 鍵 。在Monitor 開始將記憶體中的資料錄到磁帶上之前，它會先存放 10 秒鐘長度的 leader tone 在磁帶上，以便使磁帶的轉速穩定下來。Monitor 錄完後會發出“beep！”聲，此乃通知我們停止磁帶機。在操作完這些動作後，我們即可以取出磁帶，並且在上面標示該磁帶所錄製的資料及它在記憶體中的位置。

（例）

```
*0:FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6
   00 4C 02 00 60

*0.14

0000- FF AD AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60
*0.14W
```

（說明）①首先更改自位址 0000 開始的 21 個位址之內容。

②鍵入“0.14”，即將更改的資料 dump 出來看是否已更改了。

③鍵入“0.14W”，表示將這些資料錄到磁帶上去。

一般存放 4096 個記憶體中位置的內容到磁帶上大約需時 35 秒鐘，其中包含了正式錄製前的 10 秒鐘之 leader tone ，也就是它每秒鐘的錄製速度為 1,350 個位元（ bit ）。在WRITE 命令做完錄製資料後，它會將一個特別的值存放在記憶體中然後再將該值錄在磁帶上，這個特別值稱為 checksum ，它是磁帶上所錄製資料的部份和，其功用是準備做READ 命令時用來檢查READ 命令是否執行正確。

九、讀取磁帶上的資料

上節中討論到，使用Monitor的WRITE命令來執行將記憶體中的資料錄製到磁帶上去，本節中則要討論如何使用Monitor 的READ命令來執行將磁帶上的資料存回記憶體中去。在將磁帶上的資料存回記憶體中時，該資料可以儲存在記憶體中的任意位置上而不必儲存在它原來的位置，要注意的是該資料的長度是否一致，否則會漏失部份資料。與WRITE 命令相似，READ 命令的格式爲：

{起始位址} . {最終位址} R

與操作WRITE 命令的步驟一樣，在我們按 RETURN 鍵執行READ命令之前，首先啟動磁帶機讓它轉動一會兒再按 RETURN 鍵，在開始執行READ命令時，只需3秒鐘的 leader tone 即可使磁帶的轉速 穩定下來。

（例）

```
*0:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

*0.14

0000- 00 00 00 00 00 00 00 00
0008- 00 00 00 00 00 00 00 00
0010- 00 00 00 00 00
*0.14R

*0.14

0000- FF AD AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60
```

（ 說明）①首先將記憶體中準備存放磁帶資料的位置清除。

②鍵入"0.14"，表示將上述清除的部份dump 出來看是否已清除。

③鍵入"0.14R"，表示開始自磁帶上讀取資料並存放在位址0000到位址0014為止的位置上。

④再鍵入"0.14"，即檢查看看磁帶上的資料是否已正確的存放在記憶體中位址0000到位址0014上。

當Monitor 讀取磁帶上資料的工作完成之後，會比較記憶體中的checksum與磁帶上的checksum 是否相同，若不相同則表示讀取的資料有錯誤，此時會發出"deep!"聲並且會顯示"ERR"。若相同則只出現prompt "*"。

十、機器語言

機器語言（Machine language）雖然寫起來不太方便，但却是Apple 微電腦在使用上最具有效力的語言，Monitor 可以讓我們使用機器語言來從事creating, writing 及 debugging 等工作。

類似一般程式的設計，在程式中我們可以使用十六進位制來表示指令的操作元（opcodes）及運算元（operands），並且可以做上述各節所提到的功能。最重要的是，當我們要從事機器語言的程式設計時，必須使用GO 命令。即在進入Monitor時要鍵入 [G]，此時Monitor 會交由6502微處理機（microprocessor）來開始執行機器語言，而Monitor 則視此機器語言程式為一副程式（subroutine），當機器語言程式執行完成時，必須再執行一個RTS 命令（return from subroutine），此時控制會再交還給Monitor。

在設計機器語言程式時，我們可以將Monitor 中的副程式呼叫（call）出來使用，下例即是經由此類副程式而執行將英文字母"A"到

"Z"顯示出來。

(例)

```
*300:A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60

*300.30C

0300- A9 C1 20 ED FD 18 69 01
0308- C9 DB D0 F6 60
*300G
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

(說明)①首先將機器語言程式存放在位址0300開始的位置上。

②鍵入"300.30C",即將該程式dump出來看是否正確。

③鍵入"300G",即表示開始執行自位址0300開始的機器語言程式,結果顯示了英文字母"A"到"Z"。

上例中的機器語言程式實在不易看得出來它在做那些工作,現在我們必須使用Monitor的另一個命令LIST來將上述程式轉換成組合語言(assembly language)的形式,如下例。

```
*300L

0300-   A9 C1       LDA   #$C1
0302-   20 ED FD    JSR   $FDED
0305-   18          CLC
0306-   69 01       ADC   #$01
0308-   C9 DB       CMP   #$DB
030A-   D0 F6       BNE   $0302
030C-   60          RTS
030D-   00          BRK
030E-   00          BRK
030F-   00          BRK
0310-   00          BRK
0311-   00          BRK
0312-   00          BRK
0313-   00          BRK
0314-   00          BRK
0315-   00          BRK
0316-   00          BRK
```

（說明）例中“300L”表示自位址0300開始執行列印（LIST）命令。其中最左邊一行爲每一指令之起始位址，其次的中間一至三行是表示機器語言指令，而右邊的兩行則爲組合語言指令。

①程式中的位址是由程式計數器（program counter）來負責計算的，它是依據每一個指令的內蘊長度來決定下一個指令的位址。例如自位址0300開始是指令LDA，由於其內蘊長度爲2個位置，故程式計數器即自動由位址0300加上2而變成0302，此0302即爲下一個指令JSR的起始位址，依此類推。

②機器語言指令中的第一個位置（由二位的十六進位數構成）是屬於opcodes，第二個位置以後則爲operands。

③組合語言指令中左邊的一組3個字是屬於opcodes（如LDA，JSR等），右邊的部份則爲operands（如#$C1，$FDED等），它們是與機器語言相對應的，有關這些組合語言的指令格式及其功能，可以查閱本書的附錄。

④例中“LDA #$C1”表示將$C1存放到Accumulator中去，我們知道$C1即爲英文字母A之ASCII代碼，而存放到Accumulator中去乃準備輸出用。

⑤例中“JSR $FDED”表示跳去執行副程式COUT，即自Accumulator中輸出一個字元。

⑥例中“CLC”指令表示清除進位旗標（clear carry flag），其作用在下個指令中可以看得出來。

⑦例中“ADC #$01”指令表示將$01與carry加到Accumulator去，由於上個指令中已將carry清爲0，故此時Accumulator中之值只加$01，而由$C1變爲$C2，此指令每執行一次即會使得Accumulator之值增加1。

⑧例中"CMP #$DB"指令表示將Accumulator中之值與$DB比較，若相等則將processor status register（參閱附錄A中之PROGRAMMING MODEL）中的zero flag設定為1。主要目的在於檢查是否已將26個字母輸出，因為字母z之ASCII代碼為$DA故若Accumulator中之值為$DB時即表示已完成26個字母之輸出工作了。

⑨例中"BNE $0302"指令表示若Zero flag之值為0則跳到$0302位址（即指令"JSR $FDED"）去執行輸出字元的工作；否則即執行下一個指令。通常此類有條件的跳開指令，均與比較性的指令（如CMP）一起使用。

十一、組合語言

在上節的例題中，我們使用Monitor的LIST命令可看到組合語言程式，然而我們也可以直接將這些組合語言程式輸入Apple中。在Apple微電腦中稱此為Mini-Assembler，此乃因為它比一般的Assembler少些功能。又由於Mini-Assembler部份存放在Integer BASIC的ROM之中故一般Apple II Plus機種並不適用此Mini-Assembler，因此本節及第十二節均僅提供參考。使用Mini-Assembler首先要鍵入"F666G"，如下所示：

```
*F666G
```

緊接著會在下一列的左端出現"!"此乃表示已進入Mini-Assembler，我們可以在"!"之後開始鍵入程式，資料或Monitor的命令。有關Mini-Assembler的指令格式將附錄於本章最後。

Mini-Assembler是經由Program Counter而得知位址的，因

此在程式開始時，必須將Program Counter之值設定爲我們所設計的程式之起始位址，因此我們要先鍵入位址，接著鍵入冒號“：”及指令，此時這個第一個指令的位址即程式的起始位址將會存入Program Counter中。指令中的opcodes與operands 要以空白來隔開，並且在鍵入一個指令之後按上 RETURN 鍵到下一列去鍵入第二個指令。在按上 RETURN 鍵後，在我們所鍵入的指令之下會將該指令以機器語言形成顯示（即轉換成十六進位制），並依照Program Counter所指示的位址將該指令存入記憶體中。在下一列出現“！”等待鍵入第二個指令時，不可再鍵入位址與冒號“：”，只要先空一個位置即可直接鍵入指令了，並且在鍵完一個指令後要按上 RETURN 鍵以便存入記憶體中。

（例）

```
!300:LDX #02

0300-    A2 02        LDX    #$02
! LDA $0,X

0302-    B5 00        LDA    $00,X
! STA $10,X

0304-    95 10        STA    $10,X
! DEX

0306-    CA           DEX
! STA $C030

0307-    8D 30 C0     STA    $C030
! BPL #302
        ^
! BPL $302

030A-    10 F6        BPL    $0302
! BRK

030C-    00           BRK
```

（說明）① "LDX #02" 指令表示將 "02" 存放到暫存器 X中去，而 300即為此程式之起始位址。

② "LDA $0,X"指令表示將暫存器X之值與0相加後之值視為位址，並將此位址內之值存放到Accumulator 中去。

③ "STA $10,X" 指令表示將Accumulator 中之值存放到某位址去，而該位址為暫存器X之值與10 相加後之值。

④ "DEX"指令表示將暫存器X之值減去1。

⑤ "STA $C030"指令表示將Accumulator中之值存放到位址C030 的記憶體中。

⑥ "BPL $302"指令表示若processor status register中之Negative status（N）＝0，即執行中出現結果為正數之情形則跳到位址302 去執行（即跳到指令 "LDA $0,X" 去）。

⑦位址C030是用來使Apple的喇叭發出聲音的，這個程式只是做為測試用的，經由此程式的執行，我們可以聽到喇叭上所發出的聲音。

如果我們所鍵入的指令有錯誤時，我們將會聽到 "beep！" 聲，並且會出現 "∧" 來指示錯誤。一般發生錯誤的原因不外乎拼錯字或不合規格，像opcodes 與 operands間忘了空一個位置，不該出現文字的地方（如起始位址與operands）却鍵入文字，或是一些branch 指令像BCC，BCS，BEQ 等在執行時超出了127個位置的範圍。

若要停止Mini-Assembler的使用，只需按 RESET 鍵或是鍵入Monitor 命令 "FF6GG"而回到Monitor 中去，如下所示：

```
!$FF69G
```

當我們的組合語言程式已存放在記憶體中，我們可以使用Monitor的LIST 命令將該程式顯示出來。

（例）

```
*300L

0300-   A2 02       LDX   #$02
0302-   B5 00       LDA   $00,X
0304-   95 10       STA   $10,X
0306-   CA          DEX
0307-   8D 30 C0    STA   $C030
030A-   10 F6       BPL   $0302
030C-   00          BRK
030D-   00          BRK
030E-   00          BRK
030F-   00          BRK
0310-   00          BRK
0311-   00          BRK
0312-   00          BRK
0313-   00          BRK
0314-   00          BRK
0315-   00          BRK
0316-   00          BRK
0317-   00          BRK
0318-   00          BRK
0319-   00          BRK
```

十二、程式的除錯

由於許多的因素所造成，一個程式當中總會有錯誤（bug）發生，基於此點，Monitor 提供了一種方法能夠將每一條指令的問題找出來，稱這種方法為除錯（debug）。Monitor 的STEP 命令能將一個指令同時予以解碼（decodes），顯示（displays）與執行（execute）。另外，TRACE 命令則能快速的逐一處理程式中的指令，而當執行到BRK（force break）指令時即行停止。每次STEP 命令均導致

Monitor 去執行記憶體中某一個位置的指令，該位置係依 Program Counter 所指示而定，在取得指令之後先予以顯示然後再執行。在指令執行之後，會顯示出當時所有 6502 微處理機內部的所有暫存器（register）之內容，同時 Program Counter 則自動指到下一個指令之位置上（即 Program Counter 存放下一個指令之位址）。下面的例子是說明當我們使用過Mini - Assembler來存入程式之後，再由 STEP 命令來執行該程式之過程：

（例）

```
*300S

0300-   A2 02        LDX   #$02
 A=0A X=02 Y=D8 P=30 S=F8
*S

0302-   B5 00        LDA   $00,X
 A=0C X=02 Y=D8 P=30 S=F8
*S

0304-   95 10        STA   $10,X
 A=0C X=02 Y=D8 P=30 S=F8
*12

0012- 0C
*S

0306-   CA           DEX
 A=0C X=01 Y=D8 P=30 S=F8
*S

0307-   8D 30 C0     STA   $C030
 A=0C X=01 Y=D8 P=30 S=F8
*S

030A-   10 F6        BPL   $0302
 A=0C X=01 Y=D8 P=30 S=F8
*S

0302-   B5 00        LDA   $00,X
 A=0B X=01 Y=D8 P=30 S=F8
*S

0304-   95 10        STA   $10,X
 A=0B X=01 Y=D8 P=30 S=F8
```

（說明）本例係沿用上節中所學之例子來說明的，該組合語言程式係自位址 0300 開始存放，故鍵入“300S”表示自位址03 00開始執行 STEP 命令。我們可以看到各指令之位址及暫存器 A，X，Y，P，S 等之內容。

其中：“A”表示Accumulator 。

“X”表示 Index 暫存器。

“Y”表示 Index 暫存器。

“P”表示Processor status 暫存器。

“S”表示Stack point 。

① “LDX #$02” 指令表示將“02”存 放到暫存器X中，故顯示X＝02。

② “LDA $0,X”指令的執行產生A＝0C ，故我們可以知道位址（ 0+02 ）之值爲0C。

③ “STA $10,X”指令的執行使得位址（ 10+02 ）之值變爲0C。

④鍵入“12”表示將位址 12 之內容 dump 出來看，是否爲0C。

⑤ “DEX”指令的執行使得暫存器X之值減少1，即由原先的02變爲01，此時因係正數故Processor status 暫存器之值仍不變（ $30 等於二進位的 00110000 ）。

⑥ “STA $C030” 指令的執行並不改變暫存器的值。

⑦ “BPL $302”指令的執行，當Processor status 暫存器，中之值爲“30”時，由於其中的第七個位元（ bit ）爲0（即00110000的最左邊一個位元），故乃跳到位址302去執行。

TRACE 命令與STEP命令不同的地方在於STEP命令 一次只

執行一個指令，而TRACE 命令一次將全部程式執行完成，即只有在我們按 RESET 鍵或在執行完後碰到BRK 指令時才會停止。若TRACE 命令在執行一個程式的最後所碰到的不是BRK指令而是RTS (Return from subroutine)指令時，會造成不停的執行，除非我們按 RESET 鍵來停止TRACE 命令的執行。

(例)

```
*T

0306-    CA              DEX
 A=0B X=00 Y=D8 P=32 S=F8
0307-    8D 30 C0        STA    $C030
 A=0B X=00 Y=D8 P=32 S=F8
030A-    10 F6           BPL    $0302
 A=0B X=00 Y=D8 P=32 S=F8
0302-    B5 00           LDA    $00,X
 A=0A X=00 Y=D8 P=30 S=F8
0304-    95 10           STA    $10,X
 A=0A X=00 Y=D8 P=30 S=F8
0306-    CA              DEX
 A=0A X=FF Y=D8 P=B0 S=F8
0307-    8D 30 C0        STA    $C030
 A=0A X=FF Y=D8 P=B0 S=F8
030A-    10 F6           BPL    $0302
 A=0A X=FF Y=D8 P=B0 S=F8
030C-    00              BRK
030C-     A=0A X=FF Y=D8 P=B0 S=F8
*
```

(說明)本例係接上例來繼續執行的，即在STEP命令執行完位址 0304的指令後，鍵入“T”來執行TRACE命令。

十三、檢查及更改暫存器

在上節中，我們使用STEP及TRACE 命令時能夠在每一個指令執行之後，看到6502微處理機內部暫存器的內容。本節將討論我們在使用STEP、TRACE及GO命令來執行時，如何檢查及更改這些暫存器的內容。

Monitor 保留了五個記憶體位置給6502微處理機的五個暫存

器（A，X，Y，P及S）來使用，Monitor 的 EXAMINE命令是經由鍵入 CTRLE 來告訴Monitor 去顯示這些暫存器的內容，即將這五個暫存器的起始位址存到 next changeable location中，如此即可顯示這五個暫存器的內容。如果我們想要改變這些暫存器的內容時，只要先鍵入冒號“:”然後逐項鍵入暫存器的新資料即可。若在更改暫存器內容後，我們接著執行Monitor 的GO, STEP 或 TRACE 命令時，Monitor 會在執行指令之前，先將新的資料存放於此五個暫存器內。例如：

（例）

```
*CTRL E

 A=0A  X=FF  Y=D8  P=B0  S=F8
*:B0  02

*CTRL E

 A=B0  X=02  Y=D8  P=B0  S=F8
*306S

0306-    CA             DEX
 A=B0  X=01  Y=D8  P=30  S=F8
*S

0307-    8D 30 C0       STA    $C030
 A=B0  X=01  Y=D8  P=30  S=F8
*S

030A-    10 F6          BPL    $0302
 A=B0  X=01  Y=D8  P=30  S=F8
```

（說明）在每個 prompt “*”之後，均係一個動作。

①首先鍵入 CTRL E，我們可以看到A，X，Y，P及S等五個暫存器之內容。

②鍵入“:B0 02”表示將“B0”存入暫存器A中，而“02”則存入暫存器X中。

③再鍵入 CTRL E 時，我們可以看到暫存器A與X中之值

均已改變爲新的值了。

④當鍵入"306S"時，由於位址306乃是執行"DEX"指令，故暫存器X中之值即由"02"減爲"01"，其他的四個暫存器均不改變。

十四、其他的MONITOR命令

(一)在第二章中我們曾經提到Monitor的COUT副程式，在這兒將介紹以INVERSE命令來控制畫面顯示黑色字體（白底）及以NORMAL命令來控制畫面恢復原來的白色字體（黑底）。

（例）

```
*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*I

*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*N

*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*
```

(說明) 本例子中鍵入"I"表示使畫面以Inverse型式來顯示，即記憶體中位址0000到位址000F中的資料將以黑字白底顯示。當我們鍵入"N"時，則表示執行NORMAL命令，而使畫面恢復以原來的白字黑底來顯示。

(二)當我們鍵入 CTRL B 時，則會離開Monitor系統而讓Apple微電腦上ROM中的Apple Integer BASIC或Applesoft Ⅱ BASIC進入。此時係視爲重新進入BASIC ，所以我們原先所存放的程式或變數（ variables ）均被清除。若我們原先是由BASIC進入Monitor中而今欲由Monitor 再回到BASIC去，倒不如鍵入 CTRL C ，因爲這樣的話我們原來的程式及變數均會原封不動的保留著。另外，如果擁有Apple Disk Operation System（ DOS ）的話，則可以直接鍵入"3D0G"命令而回到原來所使用的語言中去，如此的話我們原來的程式與變數均同樣的保持原封不動。

(三)PRINT 命令係要求Monitor 將輸出的工作由螢光幕改由其他週邊設備來執行，其格式如下：

{插座（ slot ）編號} CTRL P

其中插座編號（ slot number ）係Apple微電腦裡面電路板後方編號 1 到 7 的插座，此時該插座上應該插有介面板（ interface card ）。例如我們要將執行的結果輸出到磁碟機（ disk driver ）上，則磁碟機的介面板要插在第 6 號插座上，此時先鍵入 6 再鍵入 CTRL P ，Monitor 便會將資料輸出到磁碟片（ diskette ）上存錄起來。若是我們所鍵入的插座編號上沒有介面板，那麼會導致機器失去控制，我們的程式及變數有可能會被清除掉。若是我們鍵入的插座編號爲 0，那麼我們的結果將會在螢光幕上顯示。

(四)KEYBOARD 命令係要求Monitor 將輸入的工作由鍵盤（ Key-board）的輸入改由其他的週邊設備來輸入。其格式如下：

{插座編號} CTRL K

與PRINT 命令一樣，若鍵入的插座編號爲0時，即表示仍由鍵盤來執行輸入的工作。

(五)Monitor 亦可以執行簡單的十六進位（ hexadacimal ）之加減運算，其格式如下：

{數值} + {數值}
{數值} + {數值}

（ 例 ）

```
*20+13
=33
*4A-C
=3E
*FF+4
=03
*3-4
=FF
```

由上面的例子中，我們可以看到簡單的兩個十六進位數值之加減運算十分地便利，其結果亦直接地顯示出來。

十五、MONITOR 的特殊用法

(一)我們在此將介紹如何在同一列上連續的使用許多的Monitor 命令，其規定如下：

①每個命令之間需以空白（ space ）來加以隔開。

②每列最多不可超過254個字的長度。

③使用到STORE 命令（ ： ）時，應以單一字的命令（如：N，I ）來隔開STORE 命令與其他命令。這些單一字命令的使用必須不會使整個執行發生資料上的改變，比如說NORMAL（N ）命令只是控制畫面而已。一般命令均是以位址開頭，但是由於冒

號“：”之後為一連串的資料，若是不以單一字的命令來加以隔開則Monitor會將下一個命令之位址視為STORE 命令中的一項資料，若不以單一字命令來隔開是會發生錯誤的。

（例）

```
*300.307 300:18 69 1 N 300.302

0300- B0 02 FF FF FF FF FF FF
0300- 18 69 01
```

（說明）①“300.307”表示將位址0300到位址0307 dump出來。

②“300：18 69 1”表示自位址0300開始存放“18”，“69”與“1”，由於其後緊接著的是一個單一字的命令“N”，故只更改三項資料。

③“300.302”表示將位址0300到位址0302 dump出來看，是否更改為“18”，“69”及“1”。

④一般單一字的命令可以不必用空白來隔開，例如：L，S，I，N等。如果在執行中出現了不合法的字時，即既非十六進位數值亦非單一字的命令時則會發出“beep！”聲，並停止執行以後的命令。

㈡接著要討論的是MOVE命令的特殊用法。我們可以將一些資料重複的存放在一個記憶體區域內，首先將要重複存放的資料先存起來：

```
*300:11 22 33
```

記住起始位址0300與重複的資料個數3，再使用下列MOVE命令的格式：

{起始位址＋資料個數} < {起始位址}　　{最終位址} M

也就是說，資料 11 ， 22 ， 33 要重複地存放在位址 0300 到位址 032D 之中，如下例：

（例）

```
*303<300.32DM

*300.32F

0300- 11 22 33 11 22 33 11 22
0308- 33 11 22 33 11 22 33 11
0310- 22 33 11 22 33 11 22 33
0318- 11 22 33 11 22 33 11 22
0320- 33 11 22 33 11 22 33 11
0328- 22 33 11 22 33 11 22 33
```

㈢同樣的，我們也可以將上述方法使用在 VERIFY 命令中，用來檢查 MOVE 命令的執行是否正確。

（例）

```
*300:0

*301<300.31FM

*301<300.31FV

*304:02

*301<300.31FV

0303-00 (02)
0304-02 (00)
```

（說明）①由於位址0300存放“0”，故“301<300.31FM”命令的執行，將使得位址301到位址320之內容均爲“0”

②由於將“02”存放於位址0304，故“301<300.31FV”命令的執行，將使得位址0303與位址0304比較時發現不同而顯示出位址0303之位址與內容“00”以及位址0304之內容“02”。同樣的，當比較位址0304與位址0305時亦發現不同而顯示位址0304之位址與內容“02”以及位址0305之內容“00”。

㈣另外一種特殊的方法相當於循環圈（loop），是用來重複地逐一顯示全部或一部份記憶體區域中的資料，其規定如下：

①須以單一字的命令作爲輸入路線（input line）的開端。

②須以“34:n”作爲輸入路線的結尾，其中n是一個十六進位數值，用來表示循環圈計算循環次數的起始值。

③欲停止此循環圈的執行時，須按 RESET 鍵。

```
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
030
•
```

（說明）例中以單一字的命令“N”爲開頭，並以n＝0定爲循環圈的起始值，即重複地逐一dump位址0300與位址0302，

而以第一次 dump 當做循環圈的第 0 次循環，第二次 dump即為循環的第 1 次循環。

十六、增加MONITOR的命令

本節主要在介紹如何建立使用者自己的Monitr 命令。由於當我們鍵入 CTRL Y 時，會使得Monitor 跳到記憶體中的 $3F8 位置去執行。我們只要在此位置內存放一個 JMP 指令，即可使Monitor 跳到我們自己設計的程式去執行工作，在此程式中我們可以設定Monitor中所有暫存器之值與輸入路線(input line)在下面的例子中，我們先設計好程式之後，當鍵入 CTRL Y 時即進入該程式並且等待繼續鍵入資料而直接執行。

(例)

```
*F666G

!300:LDY $34

0300-   A4 34       LDY   $34
! LDA 200,Y

0302-   B9 00 02    LDA   $0200,Y
! JSR FDED

0305-   20 ED FD    JSR   $FDED
! INY

0308-   C8          INY
! CMP #$8D

0309-   C9 8D       CMP   #$8D
! BNE 302

030B-   D0 F5       BNE   $0302
! JMP $FF69
```

```
030D-    4C 69 FF     JMP   $FF69
!3F8:JMP $300

03F8-    4C 00 03     JMP   $0300
!$FF69G

*THIS IS A TEST
THIS IS A TEST
```

（說明）①首先鍵入“F666G”而進到Mini-Assembler中去。

②鍵入“300:LDY $34”係表示程式之起始位址爲0300。而“LDY $34”指令則表示將位址$34中之值存放到暫存器Y去。此指令的眞正目的爲set mode，即在宣告此程式之執行，在Apple Ⅱ System Monitor 中位址$34內存放的是YSAV副程式。

③“LDA 200,Y”指令表示將暫存器Y中之值與200相加後，以此值爲位址將此位址中之值存放於Accumulator。由於位址$200係輸入緩衝器（input buffer）之位置，故此指令係將由輸入路線所輸入之資料存放於Accumulator中準備輸出。

④“JSR FDED”指令表示跳到CSW副程式去執行輸出的工作。

⑤“INY”指令表示將暫存器Y中之值加上1。

⑥“CMP #$8D”指令表示將Accumulator中之值與“$8D”比較，由於“$8D”係RETURN之代碼，故此指令之功用乃在於檢查輸入資料是否告一段落。

⑦“BNE 302”指令之功用是在輸入資料未結束時，跳回

位址0302去繼續執行輸入工作。

⑧"JMP $FF69"指令係表示回到Monitor去執行。

⑨"3F8：JMP $300 "指令表示位址$3F8中存放著跳到位址$300的程式去執行之指令。

在此例中我們鍵入 CTRL Y 後，Monitor會跳到位址$3F8去執行，即跳到位址$300的該程式去。我們由鍵盤上鍵入"THIS IS A TEST"時，螢光幕上會顯示這個鍵入的資料。因此這個示範的新增加的Monitor命令，其功能爲將自鍵盤鍵入的資料在螢光幕上輸出。

SUMMARY OF MONITOR COMMANDS

Summary of Monitor Commands.

Examining Memory.

{adrs}	Examines the value contained in one location.
{adrs1}.{adrs2}	Displays the values contained in all locations between {adrs1} and {adrs2}.
RETURN	Displays the values in up to eight locations following the last opened location.

Changing the Contents of Memory.

{adrs}:{val} {val} ...	Stores the values in consecutive memory locations starting at {adrs}.
:{val} {val} ...	Stores values in memory starting at the next changeable location.

Moving and Comparing.

{dest}<{start}.{end}M	Copies the values in the range {start}.{end} into the range beginning at {dest}.
{dest}<{start}.{end}V	Compares the values in the range {start}.{end} to those in the range beginning at {dest}.

Saving and Loading via Tape.

{start}.{end}W	Writes the values in the memory range {start}.{end} onto tape, preceded by a ten-second leader.
{start}.{end}R	Reads values from tape, storing them in memory beginning at {start} and stopping at {end}. Prints "ERR" if an error occurs.

Running and Listing Programs.

{adrs}G	Transfers control to the machine language program beginning at {adrs}.
{adrs}L	Disassembles and displays 20 instructions, starting at {adrs}. Subsequent L's will display 20 more instructions each.

The Mini-Assembler

F666G	Invoke the Mini-Assembler.*
${command}	Execute a Monitor command from the Mini-Assembler.
$FF69G	Leave the Mini-Assembler.
{adrs} S	Disassemble, display, and execute the instruction at {adrs}, and display the contents of the 65Ø2's internal registers. Subsequent S's will display and execute successive instructions.**
{adrs} T	Step infinitely. The TRACE command stops only when it executes a BRK instruction or when you press RESET.**
CTRL E	Display the contents of the 65Ø2's registers.

Miscellaneous.

I	Set Inverse display mode.
N	Set Normal display mode.
CTRL B	Enter the language currently installed in the Apple's ROM.
CTRL C	Reenter the language currently installed in the Apple's ROM.
{val}+{val}	Add the two values and print the result.
{val}−{val}	Subtract the second value from the first and print the result.
{slot} CTRL P	Divert output to the device whose interface card is in slot number {slot}. If {slot}=Ø, then route output to the Apple's screen.
{slot} CTRL K	Accept input from the device whose interface card is in slot number {slot}. If {slot}=Ø, then accept input from the Apple's keyboard.
CTRL Y	Jump to the machine language subroutine at location $3F8.

* Not available in the Apple II Plus.
** Not available in the Autostart ROM.

SOME USEFUL MONITOR SUBROUTINES

Here is a list of some useful subroutines in the Apple's Monitor and Autostart ROMs. To use these subroutines from machine language programs, load the proper memory locations or 65Ø2 registers as required by the subroutine and execute a JSR to the subroutine's starting address. It will perform the function and return with the 65Ø2's registers set as described.

$FDED COUT Output a character

COUT is the standard character output subroutine. The character to be output should be in the accumulator. COUT calls the current character output subroutine whose address is stored in CSW (locations $36 and $37), usually COUT1 (see below).

$FDFØ COUT1 Output to screen

COUT1 displays the character in the accumulator on the Apple's screen at the current output cursor position and advances the output cursor. It places the character using the setting of the Normal/Inverse location. It handles the control characters RETURN, linefeed, and bell. It returns with all registers intact.

$FE8Ø SETINV Set Inverse mode

Sets Inverse video mode for COUT1. All output characters will be displayed as black dots on a white background. The Y register is set to $3F, all others are unchanged.

$FE84 SETNORM Set Normal mode

Sets Normal video mode for COUT1. All output characters wwill be displayed as white dots on a black background. The Y register is set to $FF, all others are unchanged.

$FD8E CROUT Generate a RETURN

CROUT sends a RETURN character to the current output device.

$FD8B CROUT1 RETURN with clear

CROUT1 clears the screen from the current cursor position to the edge of the text window, then calls CROUT.

$FDDA PRBYTE Print a hexadecimal byte

This subroutine outputs the contents of the accumulator in hexadecimal on the current output device. The contents of the accumulator are scrambled.

$FDE3 PRHEX Print a hexadecimal digit

This subroutine outputs the lower nybble of the accumulator as a single hexadecimal digit. The contents of the accumulator are scrambled.

$F941 PRNTAX Print A and X in hexadecimal

This outputs the contents of the A and X reisters as a four-digit hexadecimal value. The accumulator contains the first byte output, the X register contains the second. The contents of the

accumulator are usually scrambled.

$F948 PRBLNK Print 3 spaces

Outputs three blank spaces to the standard output device. Upon exit, the accumulator usually contains $A0, the X register contains 0.

$F94A PRBL2 Print many blank spaces

This subroutine outputs from 1 to 256 blanks to the standard output device. Upon entry, the X register should contain the number of blanks to be output. If X=$00, then PRBL2 will output 256 blanks.

$FF3A BELL Output a "bell" character

This subroutine sends a bell (CTRL G) character to the current output device. It leaves the accumulator holding $87.

$FBDD BELL1 Beep the Apple's speaker

This subroutine beeps the Apple's speaker for .1 second at 1KHz. It scrambles the A and X registers.

$FD0C RDKEY Get an input character

This is the standard character input subroutine. It places a flashing input cursor on the screen at the position of the output cursor and jumps to the current input subroutine whose address is stored in KSW (locations $38 and $39), usually KEYIN (see below).

$FD35 RDCHAR Get an input character or ESC code

RDCHAR is an alternate input subroutine which gets characters from the standard input, but also interprets the eleven escape codes (see page 34).

$FD1B KEYIN Read the Apple's keyboard

This is the keyboard input subroutine. It reads the Apple's keyboard, waits for a keypress, and randomizes the random number seed (see page 32). When it gets a keypress, it removes the flashing cursor and returns with the keycode in the accumulator.

$FD6A GETLN Get an input line with prompt

GETLN is the subroutine which gathers input lines (see page 33). Your programs can call GETLN with the proper prompt character in location $33; GETLN will return with the input line in the input buffer (beginning at location $200) and the X register holding the length of the input line.

$FD67 GETLNZ Get an input line

GETLNZ is an alternate entry point for GETLN which issues a carriage return to the standard output before falling into GETLN (see above).

$FD6F GETLN1 Get an input line, no prompt

GETLN1 is an alternate entry point for GETLN which does not issue a prompt before it gathers the input line. If, however, the user cancels the input line, either with too many backspaces or with a [CTRL X], then GETLN1 will issue the contents of location $33 as a prompt when it gets another line.

$FCA8 WAIT Delay

This subroutine delays for a specific amount of time, then returns to the program which called it. The amount of delay is specified by the contents of the accumulator. With **A** the contents of the accumulator, the delay is $\frac{1}{2}(26+27A+5A^2)$ μseconds. WAIT returns with the A register zeroed and the X and Y registers undisturbed.

$F864 SETCOL Set Low-Res Graphics color

This subroutine sets the color used for plotting on the Low-Res screen to the color passed in the accumulator. See page 17 for a table of Low-Res colors.

$F85F NEXTCOL Increment color by 3

This adds 3 to the current color used for Low-Res Graphics.

$F800 PLOT Plot a block on the Low-Res screen

This subroutine plots a single block on the Low-Res screen of the prespecified color. The block's vertical position is passed in the accumulator, its horizontal position in the Y register. PLOT returns with the accumulator scrambled, but X and Y unmolested.

$F819 HLINE Draw a horizontal line of blocks

This subroutine draws a horizontal line of blocks of the predetermined color on the Low-Res screen. You should call HLINE with the vertical coordinate of the line in the accumulator, the leftmost horizontal coordinate in the Y register, and the rightmost horizontal coordinate in location $2C. HLINE returns with A and Y scrambled, X intact.

$F828 VLINE Draw a vertical line of blocks

This subroutine draws a vertical line of blocks of the predetermined color on the Low-Res screen. You should call VLINE with the horizontal coordinate of the line in the Y register, the top vertical coordinate in the accumulator, and the bottom vertical coordinate in location $2D. VLINE will return with the accumulator scrambled.

$F832 CLRSCR Clear the entire Low-Res screen

CLRSCR clears the entire Low-resolution Graphics screen. If you call CLRSCR while the video display is in Text mode, it will fill the screen with inverse-mode "@" characters. CLRSCR destroys the contents of A and Y.

$F836 CLRTOP Clear the top of the Low-Res screen

CLRTOP is the same as CLRSCR (above), except that it clears only the top 40 rows of the screen.

$F871 SCRN Read the Low-Res screen

This subroutine returns the color of a single block on the Low-Res screen. Call it as you would call PLOT (above). The color of the block will be returned in the accumulator. No other registers are changed.

$FB1E PREAD Read a Game Controller

PREAD will return a number which represents the position of a game controller. You should pass the number of the game controller (Ø to 3) in the X register. If this number is not valid, strange things may happen. PREAD returns with a number from $ØØ to $FF in the Y register. The accumulator is scrambled.

$FF2D PRERR Print "ERR"

Sends the word "ERR", followed by a bell character, to the standard output device. The accumulator is scrambled.

$FF4A IOSAVE Save all registers

The contents of the 65Ø2's internal registers are saved in locations $45 through $49 in the order A-X-Y-P-S. The contents of A and X are changed; the decimal mode is cleared.

$FF3F IOREST Restore all registers

The contents of the 65Ø2's internal registers are loaded from locations $45 through $49.

MONITOR SPECIAL LOCATIONS

Table 14: Page Three Monitor Locations

Address:		Use:	
Decimal	Hex	Monitor ROM	Autostart ROM
1008 1009	$3F0 $3F1	None.	Holds the address of the subroutine which handles machine language "BRK" requests (normally $FA59).
1010 1011	$3F2 $3F3	None.	Soft Entry Vector.
1012	$3F4	None.	Power-up Byte.
1013 1014 1015	$3F5 $3F6 $3F7	Holds a "JuMP" instruction to the subroutine which handles Applesoft II "&" commands.* Normally $4C $58 $FF.	
1016 1017 1018	$3F8 $3F9 $3FA	Holds a "JuMP" instruction to the subroutine which handles "USER" (CTRL Y) commands.	
1019 1020 1021	$3FB $3FC $3FD	Holds a "JuMP" instruction to the subroutine which handles Non-Maskable Interrupts.	
1022 1023	$3FE $3FF	Holds the address of the subroutine which handles Interrupt ReQuests.	

* See page 123 in the **Applesoft II BASIC Reference Manual**.

MINI-ASSEMBLER INSTRUCTION FORMATS

The Apple Mini-Assembler recognizes 56 mnemonics and 13 addressing formats used in 6502 Assembly language programming. The mnemonics are standard, as used in the MOS **Technology/Synertek 6500 Programming Manual** (Apple part number A2L0003), but the addressing formats are different. Here are the Apple standard address mode formats for 6502 Assembly Language:

Table 15: Mini-Assembler Address Formats

Mode:	Format:
Accumulator	None.
Immediate	#${value}
Absolute	${address}
Zero Page	${address}
Indexed Zero Page	${address},X ${address},Y
Indexed Absolute	${address},X ${address},Y
Implied	None.
Relative	${address}
Indexed Indirect	(${address},X)
Indirect Indexed	(${address}),Y
Absolute Indirect	(${address})

An {address} consists of one or more hexadecimal digits. The Mini-Assembler interprets addresses in the same manner that the Monitor does: if an address has fewer than four digits, it adds leading zeroes; if it has more than four digits, then it uses only the last four.

All dollar signs ($), signifying that the addresses are in hexadecimal notation, are ignored by the Mini-Assembler and may be omitted.

There is no syntactical distinction between the Absolute and Zero Page addressing modes. If you give an instruction to the Mini-Assembler which can be used in both Absolute and Zero-Page mode, then the Mini-Assembler will assemble that instruction in Absolute mode if the operand for that instruction is greater than $FF, and it will assemble that instruction in Zero Page mode if the operand for that instruction is less than $0100.

Instructions with the Accumulator and Implied addressing modes need no operand.

Branch instructions, which use the Relative addressing mode, require the *target address* of the branch. The Mini-Assembler will automatically figure out the relative distance to use in the instruction. If the target address is more than 127 locations distant from the instruction, then the Mini-Assembler wil sound a "beep", place a circumfex (^) under the target address, and ignore the line.

If you give the Mini-Assembler the mnemonic for an instruction and an operand, and the addressing mode of the operand cannot be used with the instruction you entered, then the Mini-Assembler will not accept the line.

第四章　記憶體的組織

本章所討論的是有關記憶體的整個結構。Apple 的 6502 微處理機（ microprocessor ）能夠直接的處理 65536 個記憶體位置，如果我們將 Apple 的記憶體當做一本書來看的話，那麼在這本書上共計有 256頁而每頁上則擁有 256個記憶體位置，例如在第 30 頁（$30）上的 256個記憶體位置是由編號 $3000 到 $30FF；我們之所以用四位的十六進位制來對每一個記憶體位置編號，此乃因為 6502 微處理機是以兩個位元組（ two eight-bit bytes ）來做為所有記憶體位置的位址，所以我們正好將這兩個位元組中的前一個位元組當做頁數（一個位元組可表示兩個數字）並且將後一個位元組當做每頁中的位置。

Apple 的 256頁記憶體位置可以區分為三個部份：隨機取存記憶體（ Random Access Memory ，簡稱 RAM ）唯讀記憶體（ Read Only Memory ，簡稱為 ROM ），輸入／輸出位置（ I／O location ）。不同部份的記憶體擁有不同的功能，今將 Apple 的記憶區配置圖示如下：

System Memory Map		
Page Number: Decimal	Hex	
Ø	$ØØ	RAM (48K)
1	$Ø1	
2	$Ø2	
.	.	
.	.	
.	.	
19Ø	$BE	
191	$BF	
192	$CØ	I/O (2K)
193	$C1	
.	.	
.	.	
.	.	
198	$C6	
199	$C7	
2ØØ	$C8	I/O ROM (2K)
2Ø1	$C9	
.	.	
.	.	
.	.	
2Ø6	$CE	
2Ø7	$CF	
2Ø8	$DØ	ROM (12K)
2Ø9	$D1	
.	.	
.	.	
.	.	
254	$FE	
255	$FF	

圖　　五

一、隨機取存記憶體的儲存區

Apple 的隨機取存記憶體（ RAM ）部份，我們由記憶區配置圖中可以看到是自第 0 頁開始到第 191頁（即十六進位制的第 $BF頁）；Apple 的主電路板（ main circuit board ）上可以擁有4K（4，096 bytes ）到 48K （ 49，152 bytes ）的RAM視實際需要而定；另外，我們也可以藉著增設Apple 語 言 面 板（ Apple Language Card-零件編號： A2B0006 ）而使Apple 的RAM 擴增到 64K（65，536 bytes ），增加的 16K RAM取代了Apple 原來唯讀記憶體（ ROM）

的部份(即佔用了記憶區配置圖中第 $D0 頁到第 $FF 頁的位置),此增加的 16K RAM中多出來的 4K (原來的ROM 只佔有 12K的記憶體位置,故增加的 16K RAM中多出了 4K的 RAM)則與 16K RAM 中的另外 4K RAM 共用記憶區配置圖中第 $D000 位置到第 $DFFF。

大部份Apple 中的RAM 記憶體位置均能讓我們用來儲存程式與資料,但RAM 記憶體中亦保留了一些位置以便儲存系統監督程式(System Monitor)、各種的程式語言與其他的系統機能(System function),今將RAM 記憶區位置的使用情形圖示如下:

Table 16: RAM Organization and Usage

Page Number: Decimal	Hex	Used For:	
0	$00	System Programs	
1	$01	System Stack	
2	$02	GETLN Input Buffer	
3	$03	Monitor Vector Locations	
4 5 6 7	$04 $05 $06 $07	Text and Lo-Res Graphics Primary Page Storage	
8 9 10 11	$08 $09 $0A $0B	Text and Lo-Res Graphics Secondary Page Storage	FREE
12 through 31	$0C $1F		
32 through 63	$20 $3F	Hi-Res Graphics Primary Page Storage	RAM
64 through 95	$40 $5F	Hi-Res Graphics Secondary Page Storage	
96 through 191	$60 $BF		

表 十 六

我們將表一16 中之部份使用情況說明於下：

㈠ Zero Page

基於Apple 6502微處理機結構上的需要，乃將機器語言程式(machine language programs)固定的儲存在第$00頁內；其中系統監督程式佔用了第$00頁內的大約20 個位置，Apple Integer BASIC 也佔用了另外的一部份位置，剩餘部份則被Applesoft Ⅱ BASIC與 Apple磁碟作業系統(Disk Operation System，簡稱爲DOS)所佔用。我們在附錄中亦表列了Zero Page 的使用情形，請參考著看。

㈡ Page One

Apple的6502微處理機保留了Page 1上的256 個bytes做爲疊表("stack")用，雖然在通常的使用情況下，每次均祇需要其中不到半數的位置，但是由於我們不易判斷出到底其中那些部份被使用著而那些部份又空著，所以最好不要試著去使用Page 1來存放資料，以免發生問題。

㈢ Page Two

我們在前面討論過，副程式GETLN是爲了取得輸入路綫(input line)而Page Two即是用來做GETLN的輸入緩衝器(input buffer)，當然如果我們能確定不會輸入長串的資料，即RAM記憶體Page 2 中的後面位置(指位址較高的部份)不會使用到，那麼就可以安全地將資料暫時儲存在Page 2的後面位置上。

㈣ Page Three

Apple的Monitor ROM(包括Autostart及Monitor本身的ROM)佔用著Page 3中後面部份較高位址的16 個位置，即由位址$03F0到位址$03FF(相當於十進位址1008到位址1023。

㈤ Four Through Seven

在這區域內的1024-byte 記憶體位置是用來顯示Text 及低解析度圖示的首頁（Low-Resolution Graphics Primary Page），因此不能做爲儲存資料用。其中的64 個位置並不在螢光幕上顯示出來，它們是被保留著給週邊面板（Peripheral cards）使用。（關於peripheral cards我們將在下一章中加以說明）

二、隨機取存記憶體的結構

如果你擁有的是Apple Ⅱ Plus 機器，則本節可略去不看，因爲我們所將要討論的問題在Apple Ⅱ Plus 上均已修正而予以簡化了。

Apple 的RAM 記憶體部份是由8個到24 個的積體電路（integrated circuit ，簡稱爲IC）所組成，這些IC均插在Apple主電路板的三列插孔上，每列插孔均有8個插孔可供給8個IC（或稱爲晶片）來使用，這8個晶片則可以擁有總共4096-byte(4K）或16,384-byte(16K）的記憶體位置。4K的晶片屬於Mostek "4096"型類，其零件編號爲"MK4096"或MCM6604"；16K的晶片則屬於"4116"型類，其零件編號爲"MK 4116"或"UPD 4160"；在裝設這些晶片時，同一列上的晶片均應屬於同一型類，但不同列則不必屬於同一型類。

若一列上的8個IC構成16K的RAM（即16,384 eight-bit bytes 的RAM），最左邊的一個IC存放的是該16K RAM 記憶體區域內每個記憶體單位（byte）的最低數位（least significant）而最右邊的一個IC 存放的則是該記憶體區域內每個記憶體單位的最高數位（most significant）。在電路板上的三列RAM 當中，最前方的一列是由記憶體位址0開始，然後是第二列，第三列，其位址依次增高。

我們也可以藉著在Apple 電路板左邊的三個IC 插孔中插入

Memory Configuration Blocks 來設定Apple的記憶體大小與分配，Memory Configuration Blocks 是一個具有14-釘腳的IC，其內沒有晶片而祇有三條jumper wire，這些jumper wire會將每列上的RAM晶片導入Apple記憶體中的特殊位置，這三個Memory Contiguration Blocks是同一導向的。Apple公司供應這些Configu-ration Blocks給客戶使用，如果您想擴充Apple記憶體，而這些RAM不是購自Apple公司時，那麼您必須自己設法裝配您自己的Configuration Blocks。

Apple具有9種不同的RAM記憶體配置，這9種不同的配置是由於在三列插孔上不同個數的4K與16K RAM晶片組合所造成的；在圖一6中我們可以看到，橫坐標所標示的是9種不同配置的記憶體大小，縱坐標所標示的是記憶體的位址；圖中所畫的9個直行長方塊各表示配置情形，譬如最右一行中只有一個4K的長方塊即表示三列RAM插孔中只有其中第一列插有RAM chip，並且其記憶體大小爲4K。（圖六見下頁）

上述的14-釘腳IC中，右上方編號12，13，14的三支表示Apple主電路板上的三列RAM晶片，在Configuration Blocks內的這三支釘脚均須以jumper wire與其他的11支釘脚連接，其連接情形全視記憶體中的RAM chip之分配來決定。今將這14釘脚圖示如下：

4K range $0000-$0FFF	1 ○	14	Frontmost row ("C")
4K range $1000-$1FFF	2	13	Middle row ("D")
4K range $2000-$2FFF	3	12	Backmost row ("E")
4K range $3000-$3FFF	4	11	No connection.
4K range $4000-$4FFF	5	10	16K range $0000-$3FFF
4K range $5000-$5FFF	6	9	16K range $4000-$7FFF
4K range $8000-$8FFF	7	8	16K range $8000-$BFFF

圖 七 Memory Configuration Block Pinouts

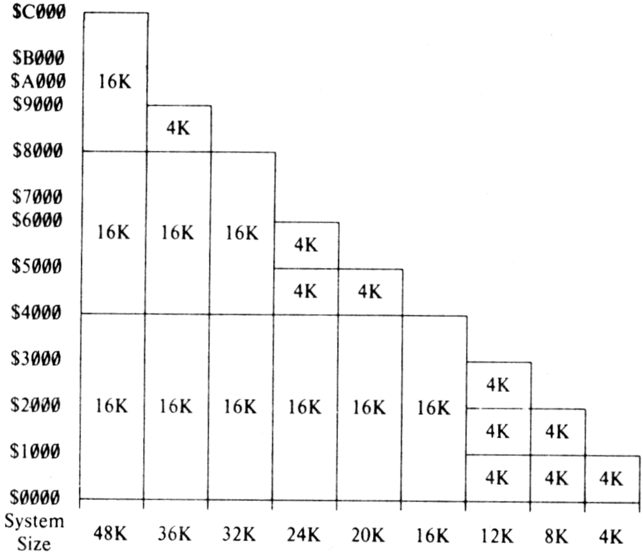


圖 六 Memory Configurations

如果某列上的八個RAM晶片擁有16K，首先我們需要看是那一列，設若是最前面的一列（釘脚編號爲14 ）則因其位址係由0開始，所以我們要將編號14 與編號10 的兩支釘脚以 jumper wire 互相連接；如果最前面的一列上八個RAM晶片只有4K ，那麼我們即要將編號14 與編號1的兩支釘脚以 jumper wire互相連接。要特別注意，絕對不能將只有4K的RAM晶片去連接16K range之釘脚，也不能將擁有16K RAM晶片的列去連接4K range的釘脚。不要漏接其中任何一列的釘脚在range釘脚上，也不要將其中兩列的釘脚連接在同一個range的釘脚上。

我們要確定自己所擁有的記憶體配置是屬於何種，設若我們的3組晶片（每組有八個chip）中，每組只擁有4K ，但我們却希望能夠

使用高解析度的圖示（ High- Resolution Graphics ），由於記憶體的大小受限制，此時只能使用 primary page ，那麼我們需將其中的一組 4K chip 定在位址 0 開始的最前面一列，即連接編號 14 與編號 1 的釘脚（ 4K range $0000-$0FFF)；另將剩餘的 2 組 4K 晶片定在 High - Resolution Graphics Primary Page的起始位址 $2000 上，即連接編號 13 與編號 3 的釘脚（ 4K range $2000-$2FFF) 以及連接編號 12 與編號 4 的釘脚（ 4K range $3000-$3FFF ）；如此我們就可以有 4K bytes 的 RAM 記憶體以及 8K bytes 的 RAM 做為繪圖使用的緩衝器（ Picture buffer ）。

自面對 Apple 前方的角度看，裝在 Apple 中的 Configuration blocks 前面邊緣部份有白點的地方即代表編號 1 的釘脚。

Apple 的 Revision 0 電路板在擁有 20K 或 24K 的 RAM 時，由於其中記憶體配置內的 8K range （由位置 $4000 到 $5FFF) 複寫在 range $6000 到 $7FFF 的記憶體中而不論該 8K range 是否有 RAM 晶片，因此雖然只有 20K 或 24K 的 RAM 但却似乎擁有 24K 或 36K，當然這多出來的 RAM 純係想像而已，這個問題在 Apple 的 Revision 1 電路板上已經加以改善了。

三、唯讀記憶體的儲存區

在一般的情形中，Apple 的主電路板上能擁有 2K 到 12K 的唯讀記憶體（ ROM ），這些 ROM 記憶體包括了系統監督程式（System Monitor ），BASIC 語言及其他系統程式與共用程式，或可包含 Apple 的 programmer's Aid #1 ROM 之內的 pre-packaged 副程式等。

Apple 的 12K ROM 在記憶區配置圖中之起始位址為 $D000，基於適當的操作與使用 Apple 之需要，必須將某些 ROM 置於記憶體

中的最上方（即最接近位址 $D000 的位置）。當我們開啟Apple 時，微處理機（microprocess）必須有程式來使它執行動作，即微處理機會到記憶區配置圖中存放這個程式的起始位址上去執行；通常這個起始位址係固定地存放在ROM 內以便微處理機取得，而這個起始位址所對應的程式亦同樣地存放在ROM 中；這個程式的作用是將Apple 置於初始狀態（initializes）而讓我們開始操作與使用。今將ROM 的記憶區位置的使用情形圖示如下：

<table>
<tr><th colspan="4">Table 17: ROM Organization and Usage</th></tr>
<tr><th colspan="2">Page Number:
Decimal Hex</th><th colspan="2">Used By:</th></tr>
<tr><td>208
212</td><td>$D0
$D4</td><td>Programmer's Aid #1</td><td rowspan="4">Applesoft
II
BASIC</td></tr>
<tr><td>216
220</td><td>$D8
$DC</td><td></td></tr>
<tr><td>224
228
232
236
240</td><td>$E0
$E4
$E8
$EC
$F0</td><td>Integer BASIC</td></tr>
<tr><td>244</td><td>$F4</td><td>Utility Subroutines</td></tr>
<tr><td>248
252</td><td>$F8
$FC</td><td>Monitor ROM</td><td>Autostart ROM</td></tr>
</table>

表十　七

Apple 的主電路板上有六個的24-釘腳 IC 插孔用來插上ROM IC ，其中每一個插孔能插上 9316B 型號的 2K bytes（2048-byte, 8bit-byte)唯讀記憶體 IC（ROM IC），最左邊的一個位置擁有ROM 記憶區配置中最高位的位址而最右邊的一個則自最低位的位址 $D000 開始。如果ROM IC不插在此六個插孔上，那麼空出來的插孔所對應的記憶區內的值將無法知悉。

Apple Firmware面板能夠將Apple 主電路板上全部或部份的ROM改由該面板上的ROM來取代，當我們將Apple Firmware面板插入Apple 主電路板上的插座（slot）上時，如果欲以面板上的ROM取代主電路板上的ROM ，只要將面板上的開關扳到DOWN的位置並

按 RETURN 鍵即可；另外我們也可以鍵入位址 $C081（十進位制的 49281 或 -16255），有關此部份的詳細情形請參考附錄D。

四、輸入／輸出的位置

Apple 記憶區配置中有 4K（4096-byte）是做爲輸入與輸出工作的，這 4K 的記憶體是自位址 $C000（十進位制的 49152 或 -16384）到位址 $CFFF（十進位制的 53247 或 -12289），詳細情形我們將在第五章中討論。

ZERO PAGE MEMORY MAPS

Table 18: Monitor Zero Page Usage

Decimal	Hex	Ø $Ø	1 $1	2 $2	3 $3	4 $4	5 $5	6 $6	7 $7	8 $8	9 $9	1Ø $A	11 $B	12 $C	13 $D	14 $E	15 $F
Ø	$ØØ																
16	$1Ø																
32	$2Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
48	$3Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
64	$4Ø	•	•	•	•	•	•	•	•	•	•					•	•
8Ø	$5Ø	•	•	•	•	•	•										
96	$6Ø																
112	$7Ø																
128	$8Ø																
144	$9Ø																
16Ø	$AØ																
176	$BØ																
192	$CØ																
2Ø8	$DØ																
224	$EØ																
24Ø	$FØ																

Table 19: Applesoft II BASIC Zero Page Usage

Decimal	Hex	Ø $Ø	1 $1	2 $2	3 $3	4 $4	5 $5	6 $6	7 $7	8 $8	9 $9	1Ø $A	11 $B	12 $C	13 $D	14 $E	15 $F
Ø	$ØØ	•	•	•	•	•	•					•	•	•	•	•	•
16	$1Ø	•	•	•	•	•	•	•	•	•							
32	$2Ø																
48	$3Ø																
64	$4Ø																
8Ø	$5Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
96	$6Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
112	$7Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
128	$8Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
144	$9Ø	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
16Ø	$AØ	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
176	$BØ	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
192	$CØ	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
2Ø8	$DØ	•	•	•	•	•	•			•	•	•	•	•	•	•	•
224	$EØ	•	•	•		•	•	•	•	•	•	•					
24Ø	$FØ	•	•	•	•	•	•	•	•	•							

Table 20: Apple DOS 3.2 Zero Page Usage																	
Decimal		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Hex	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
0	$00																
16	$10																
32	$20							●	●			●	●	●	●	●	●
48	$30						●	●	●	●	●					●	●
64	$40	●	●	●	●	●	●	●	●	●		●	●	●	●		
80	$50																
96	$60								●	●	●	●					●
112	$70	●															
128	$80																
144	$90																
160	$A0																●
176	$B0	●															
192	$C0											●	●	●	●		
208	$D0									●							
224	$E0																
240	$F0																

Table 21: Integer BASIC Zero Page Usage																	
Decimal		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Hex	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
0	$00																
16	$10																
32	$20																
48	$30																
64	$40											●	●	●	●		
80	$50						●	●	●	●	●	●	●	●	●	●	●
96	$60	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
112	$70	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
128	$80	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
144	$90	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
160	$A0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
176	$B0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
192	$C0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
208	$D0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
224	$E0																
240	$F0																

第五章　輸入及輸出之結構

Apple 的輸入及輸出方式最主要的有二種，第一種是在 Apple 本身的電路板上來執行，第二種是以週邊設備介面板插入 Apple 的八個週邊裝備插孔上而執行；以上的二種方式均經由 Apple 的 4096 個記憶區位置與微處理機及你的程式相連繫，在本章裡將描述記憶區配置及不同的輸入／輸出控制與功能的操作使用，關於硬體綫路如何執行這些功能我們將在下章討論。

一、固定的輸入與輸出

大部份有關 Apple 固定的 I／O 設備均已簡要的在第一章（"Approaching your Apple"）中敘述過了，若你想大略地了解這些設備請看第一章。

Apple 的鍵盤直接輸入／輸出功能，在 Apple 的記憶區配置圖上是以 128 個記憶體位置來控制的，這 128 個記憶體位置是自起點位址 $C000 延伸到位址 $C07F 為止，如以十進位制表示則是由位址 49152 到 49279（或是由位址 − 16384 到 − 16257）；這128 個位置是由 27 種不同的功能所分佔，很明顯的有些功能必須有多個位置來使用，在某些例子中，最多有達16個不同的位置能夠執行同樣一種功能，而這 128 個位置又可分成下列五種形態：

(一)Data Input

在 Apple 電路板上唯一的資料輸入是一個位置（即 one byte），而該位置儲存的值則表示 Apple 內設鍵盤的訊息流動狀態，經由鍵盤輸入的訊息中之最高位數元與 Flag Input（下面會討論到）類

似，至於低位的七個數元則表示由鍵盤所鍵入的訊息之ASCII代碼。

(二) Flag Input

Apple 的大部份內設輸入位置均爲單一位元的"flags"，這些flags則位於我們所輸入每個訊息之八個數元中的最高位數元。flags只以"on"與"off"兩種值來表示，我們可以經由任何一種語言來測試flags的設定值，而高階語言則能使用"PEEK"或類似的命令來讀取flags位置中的值；如果我們讀取到的flags值等於或大於128即表示flags爲on值，如果小於128即表示flags爲off值。使用機器語言時，我們可以把flags位置內之值儲存到6502微處理機的任何一個內在暫存器上去；我們也可以依據N flag來執行跳開的動作，也就是說當flag爲on值時我們可以使用BMI指令而跳開到其他地方去執行，並且當flag爲off值時可以使用BPL指令來跳開。

像Single-Bit（Pushbotton）的輸入、卡式磁帶的輸入、鍵盤訊息的輸入及Game控制器的輸入等均屬於flag input。

(三) Strobe Outputs

像通用程式特殊指示信號（Utility Strobe）、鍵盤清除特殊指示信號（Clear Keyboard Strobe）及遊戲控制器特殊指示信號（Game Controller Strobe）等均由記憶體位置所控制，如果我們在程式裏讀取這些特別的位置，那麼與這些位置連繫的相關功能即會開始執行。例如Utility Strobe的情況中，Game輸入／輸出連接器上編號5的釘腳之電壓會由+5V降到0V（週期爲0.98微秒）然後再回復到+5V；在Clear Keyboard Strobe的情況時，鍵盤的flag input將會變成off值；又在Game Controller Strobe的情況時，所有Game Controller的flag input將會變成off值並且它的時間週期會重新開始。

我們也可以在程式中寫入這些控制位置來啟動Keyboard及Game Controller Strobe（但是無法寫入Utility Strobe的位置），如此的話我們將會得到兩個 0.98×10^{-6} 秒的電波而其間隔爲 24.43×10^{-9} 秒，原因是6502 微處理機將資料寫到記憶體位置去時首先要讀取此位置之內容然後再將此內容儲存到該位置上去；上述兩個電波在Keyboard 及Game Controller Strobe 的情況時是毫無問題的，但若是出現在Utility Strobes 時則會產生問題。

(四) Toggle Switches

另外有兩種特殊指示信號（strobe）的輸出是連接在兩段"flip-flop" 裏面，每次當我們讀取記憶體中的strobe 控制位置時，它的起伏電路（flip-flop） 會由當前的狀態跳到另一個狀態，這個跳動式開關（toggle switches）能帶動卡式磁帶的輸出及內附的擴音器；然而並沒有一種實際的方法來設定此內附的toggle switches ，因爲toggle switches 只保持其原有的狀態而不能隨意的改變，它與前面所討論過的Strobe Outputs 情形一樣， Strobe 的控制位置儲存著固定的值，經由這個值我們可以獲得一些功能的執行，所以我們只能去讀取這些控制位置不可以寫入資料到該位置上而改變了它的值，否則會產生問題。

(五) Soft Switches

軟體開關（soft switches）是一種兩個端點的開關，這兩個端點各由不同的記憶體位置來控制。如果我們指示記憶體位置到此軟體開關中的一端上，則軟體開關即會開向這一端上，如果我們指示記憶體位置到另一端上，則軟體開關即向著另一端。也就是說完全依據我們指示的記憶體位置來決定軟體開關的方向，在設定此記憶體位置時是不管它以前所存放的爲何值，也無法知道以前的記憶體位置是指示到軟體開關的那一端。我們可以很安全的存放資料到此軟體開關的控

制位置上，另外我們也知道以兩個電波的方式是比一個電波來得好。指示器（ Annunciator ）的輸出及所有的螢光幕輸出型式亦均由此軟體開關來控制。

控制著建立在機器內部的輸入／輸出功能之特殊記憶體位置其編配如下圖：

Table 22: Built-In I/O Locations

	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$C000	Keyboard Data Input															
$C010	Clear Keyboard Strobe															
$C020	Cassette Output Toggle															
$C030	Speaker Toggle															
$C040	Utility Strobe															
$C050	gr	tx	nomix	mix	pri	sec	lores	hires	an0		an1		an2		an3	
$C060	cin	pb1	pb2	pb3	gc0	gc1	gc2	gc3	repeat $C060-$C067							
$C070	Game Controller Strobe															

表 二 十 二

縮寫字詳述：

gr	設定爲GRAPHICS模式（ Set GRAPHICS mode)
nomix	設定全部以 text 或 graphics 顯示（ Set all text or graphics ）
pri	顯示主要輸出頁（ Display primary page ）
lores	顯示低解析度圖形（ Display Low-Res Graphics ）
an	指示器的輸出（ Annunciator outputs ）
gc	遊戲控制器的輸入（ Game Controller inputs ）
tx	設定TEXT模式（ Set TEXT mode ）
mix	混合 text 及 graphics 來顯示（ Mix text and graphics ）

sec	顯示次要輸出頁（ Display secondary page ）
hires	顯示高解析度圖形（ Display Hi-Res Graphics ）
pb	遊戲控制器上按鈕的輸入（ Pushbutton inputs ）
cin	磁帶機的輸入（ Cassette Input ）

二、週邊面板的輸入／輸出

在 Apple 的主要電路後部有一排（八個）插座（ slot ）或稱爲週邊連接孔（ Peripheral Connectors ）。在這八個插座中，其中有七個插座可以插入任何的爲Apple 而設計之週邊介面板（ Peripheral Interface boards）。爲了使這些週邊面板能夠簡單操作及更多的功能，在 Apple 的硬體綫路裡的記憶區配置內保留了總共280位元組的位置給這七個插座，另外有 2K 的位元組共用區給所有的週邊面板來分用。

每一個面板插座均有不同的編號，最左邊的爲0 號插座，最右邊的爲 7號插座。其中 0 號插座比較特殊，它是爲RAM、ROM或介面板的擴充（ Interface expansion ）而設計。其餘的由 1 號到 7 號插座均有特殊的控制綫路來連接，每一插座在均可在不同的時間操作及執行動作。

三、週邊面板的輸入／輸出空間

自位址 $C080開始每一個插座均給予十六個記憶體位置來做一般性的輸入及輸出用。其中， 0 號插座佔有記憶體位置 $C080 到 $C08F， 1 號插座佔有記憶體位置 $C090 到 $C09F 等依此類推。每一個週邊面板都能在需要的時候去使用這些位置，每一個週邊面板也可以決定何時聽從第 41 號釘脚（稱爲$\overline{\text{DEVICE SELECT}}$）而被

選擇到週邊連接孔去。無論何時當第41號釘脚的電壓降到 0 伏特，微處理機會呼叫被選擇的週邊面板所擁有之16個記憶體位置，這時週邊面板便可檢視其16個位址中的後面 4 個位址來決定到底16個位址中的那一個位址被呼叫（ call ）。

Table 23: Peripheral Card I/O Locations

	$Ø	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$CØ8Ø										Ø						
$CØ9Ø										1						
$CØAØ										2						
$CØBØ						Input/Output for slot number				3						
$CØCØ										4						
$CØDØ										5						
$CØEØ										6						
$CØFØ										7						

表　二　十　三

四、週邊面板的ROM空間

每一個週邊插座均保留有一頁共256個位元組的記憶體來使用，而這頁記憶體通常是 256 個位元組的唯讀記憶體（ ROM ）或可程式的ROM（ Programmable ROM或簡稱爲PROM ）。這頁的 ROM 或PROM中包含了週邊面板的設置程式（ driving programs ）或副程式，由於擁有自己的設置軟體程式，因此使得週邊介面板具有“智慧”，所以我們在使用這些介面板時不必個別的去饋入（ load ）使用程式。

這些保留給每一個週邊插座使用的一頁記憶體，我們給予頁數編號爲 $Cn，其中 n 即表示插座的編號。由於 0 號插座並沒有保留的記憶體，所以無法將任何的介面板插置在 0 號插座。當微處理機照會其

中之一的週邊面板插座在記憶體中所保留的位址時，該週邊面板插座上的第1號釘脚（稱爲$\overline{\mathrm{I/O\ SELECT}}$）會開始執行動作（即電壓會由＋5伏降到0伏）。週邊面板即利用此信號使其PROM 開始執行，並且利用其256個位元組中較低位址的8個位元組去編排PROM中每一個位元組的位址。

Table 24: Peripheral Card PROM Locations

	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
$C100										1						
$C200										2						
$C300										3						
$C400	PROM space for slot number									4						
$C500										5						
$C600										6						
$C700										7						

表 二 十 四

五、設計輸入／輸出程式時的要點

在週邊面板PROM中的程式是可以搬動的，也就是說不管這些程式將被存放於記憶區配置中的任何位置去，我們均能加以更改。必須無絕對的指示在這些程式上，這些程式同時必須能執行JuMPs 指令而不論是强迫或條件的。

當然我們可以在週邊面板的PROM裡存放一些不可搬動的副程式。這樣做的唯一損失是，這片的週邊面板將變成獨立的插座（slot-dependent）。爲了使週邊面板PROM有更多的記憶體空間可供利用，我們可以經由使副程式成爲slot-dependent而得到更多的記憶體位置。

在週邊面板PROM中的副程式所要做的第一件工作，就是存放所有在6502微處理機內部的暫存器（ registers ），在Apple的Monitor ROM中有一個副程式IOSAVE在做這一件工作。此一副程式會從記憶體位置$45開始到$49為止，依A－X－Y－P－S的順序來存放所有內部暫存器的內容，這個副程式本身則是從位址$FF4A開始存放的。另有一個相對的副程式IORESTORE，是用來將這些儲存的位置內之值（即位置$45到$49為止的內容）存回到所有的內部暫存器去。在我們執行PROM副程式的當中，我們可以呼叫此IORESTORE副程式，其起始位址為$FF3F。

大部份單一字元的輸入及輸出是經由6502微處理機的累積器（Accumulator）。像在做輸出工作時，所要顯示的字元是存放在累積器中的，同時亦設定其高位元。在做輸入工作時，我們的副程式亦須將來自輸入設備的字元經由累積器輸入，此時也設定其高位元。

在週邊面板PROM中的程式能夠經由下列一連串指令的執行而決定到底此週邊面板是插置於那一個插座中：

```
0300-   20 4A FF   JSR   $FF4A
0303-   78         SEI
0304-   20 58 FF   JSR   $FF58
0307-   BA         TSX
0308-   BD 00 01   LDA   $0100,X
030B-   8D F8 07   STA   $07F8
030E-   29 0F      AND   #$0F
0310-   A8         TAY
```

當上述一連串動作執行後，該週邊面板所插置的插座編號將會以$0n（其中，n即為插座編號）的形式儲存在6502微處理機的Y指

標暫存器（ Y index register ）內。在ROM中的此一程式能進一步的將該暫存器內之值左移 4 個位元而得到$n0值，其處理的一連串指令如下：

```
0311-   98      TYA
0312-   0A      ASL
0313-   0A      ASL
0314-   0A      ASL
0315-   0A      ASL
0316-   AA      TAX
```

程式能夠利用此以 6502 微處理機的指標位址模式儲存在 X 指標暫存器內之值去查詢那些保留給每一週邊面板的16個記憶體位置。例如指令：

```
0317-   BD 80 C0     LDA   $C080,X
```

會執行將週邊面板所使用的 I／O locations 之第一個位置內之值儲存於 6502 微處理機的累積器中。上述指令中的位址 $C080 係所有週邊插座所使用 I／O locations 中的第一個位置，在此是做爲基本位址（ Base address ）的，而位址 $C081 則是所有週邊插座所使用 I／O locations 中第二個位置的基本位址，以此類推。今列出每一週邊面板的全部十六個 I／O locations 之基本位址如下：

Table 25: I/O Location Base Addresses

Base Address	Slot 0	1	2	3	4	5	6	7
$CØ8Ø	$CØ8Ø	$CØ9Ø	$CØAØ	$CØBØ	$CØCØ	$CØDØ	$CØEØ	$CØFØ
$CØ81	$CØ81	$CØ91	$CØA1	$CØB1	$CØC1	$CØD1	$CØE1	$CØF1
$CØ82	$CØ82	$CØ92	$CØA2	$CØB2	$CØC2	$CØD2	$CØE2	$CØF2
$CØ83	$CØ83	$CØ93	$CØA3	$CØB3	$CØC3	$CØD3	$CØE3	$CØF3
$CØ84	$CØ84	$CØ94	$CØA4	$CØB4	$CØC4	$CØD4	$CØE4	$CØF4
$CØ85	$CØ85	$CØ95	$CØA5	$CØB5	$CØC5	$CØD5	$CØE5	$CØF5
$CØ86	$CØ86	$CØ96	$CØA6	$CØB6	$CØC6	$CØD6	$CØE6	$CØF6
$CØ87	$CØ87	$CØ97	$CØA7	$CØB7	$CØC7	$CØD7	$CØE7	$CØF7
$CØ88	$CØ88	$CØ98	$CØA8	$CØB8	$CØC8	$CØD8	$CØE8	$CØF8
$CØ89	$CØ89	$CØ99	$CØA9	$CØB9	$CØC9	$CØD9	$CØE9	$CØF9
$CØ8A	$CØ8A	$CØ9A	$CØAA	$CØBA	$CØCA	$CØDA	$CØEA	$CØFA
$CØ8B	$CØ8B	$CØ9B	$CØAB	$CØBB	$CØCB	$CØDB	$CØEB	$CØFB
$CØ8C	$CØ8C	$CØ9C	$CØAC	$CØBC	$CØCC	$CØDC	$CØEC	$CØFC
$CØ8D	$CØ8D	$CØ9D	$CØAD	$CØBD	$CØCD	$CØDD	$CØED	$CØFD
$CØ8E	$CØ8E	$CØ9E	$CØAE	$CØBE	$CØCE	$CØDE	$CØEE	$CØFE
$CØ8F	$CØ8F	$CØ9F	$CØAF	$CØBF	$CØCF	$CØDF	$CØEF	$CØFF
	I/O Locations							

表 二 十 五

六、週邊設備插座的SCRATCHPAD RAM

在 Apple 的ROM記憶體中保留有 8 個位置給每個週邊插座，這總共 64 個位置事實上是在記憶體的第 $04 頁到 $07 頁上，亦即是在 Text 及 Low-Resolution Graphics Video Display 的保留區域內。但是這些位置的內容並不在螢光幕上顯示，並且這些位置的內容也不因螢光幕的操作而改變。當週邊面板在執行操作時能夠利用位置來暫時儲存一些資料，這些抓來裝塡（ scratchpad ）的位置其位址如下：

Table 26: I/O Scratchpad RAM Addresses							
Base	Slot Number						
Address	1	2	3	4	5	6	7
$0478	$0479	$047A	$047B	$047C	$047D	$047E	$047F
$04F8	$04F9	$04FA	$04FB	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B	$057C	$057D	$057E	$057F
$05F8	$05F9	$05FA	$05FB	$05FC	$05FD	$05FE	$05FF
$0678	$0679	$067A	$067B	$067C	$067D	$067E	$067F
$06F8	$06F9	$06FA	$06FB	$06FC	$06FD	$06FE	$06FF
$0778	$0779	$077A	$077B	$077C	$077D	$077E	$077F
$07F8	$07F9	$07FA	$07FB	$07FC	$07FD	$07FE	$07FF

表 二 十 六

編號 0 的插座是沒有保留 scratchpad RAM 給它的，上表中的基本位址（ Base address ）是可以分給Apple DOS 3.2 及其他週邊面板使用的。在這些位置中，有些是屬於共用功能的：位置 $7F8 中存放著目前正使用中的週邊面板所插置的插座編號，而位置 $5F8 中則存放著啟動使用DOS之磁碟控制面板（ disk controller card）所在的插座編號。

在上節所討論的程式例子中，我們知道週邊面板所插置的插座編號是以 $0n 的形式儲存在 6502微處理機的Y指標暫存器中，今以另一個副程式來直接查詢八個 scratchpad 位置中的任何一個位置：

```
031A-   B9 78 04    LDA   $0478,Y
031D-   99 F8 04    STA   $04F8,Y
0320-   B9 78 05    LDA   $0578,Y
0323-   99 F8 05    STA   $05F8,Y
0326-   B9 78 06    LDA   $0678,Y
0329-   99 F8 06    STA   $06F8,Y
032C-   B9 78 07    LDA   $0778,Y
032F-   99 F8 07    STA   $07F8,Y
```

七、CSW／KSW輸出開關

位置 $36 及 $37（十進位制的 54 及 55 ）稱之爲字元輸出開關（ Character Output Switch 或簡稱爲 CSW ）。分開來說明時，稱位置 $36 爲 CSWL（ CSW Low ）而位置 $37 則稱爲 CSWH(CSW High ）。 這兩個位置中存放著使用中的 Apple 單字元輸出副程式之位址，通常這副程式（ 即 COUT ）的起始位址爲 $FDF0 。我們可以經由Monitor 的PRINTER（ CTRL P ）命令及 BASIC命令的 PR#，來改變此位置之內容爲週邊面板上 PROM 的副程式之起始位址。這兩種命令（ command ）會將位址 $Cn00 存入位置 $36及 $37 中，其中 n 爲插座編號由命令中加以指定。位址 $Cn00 乃是插座上所插置週邊面板內PROM之第一個位址，如此 Apple 每次要輸出一個字元時即可呼叫此副程式，並且此副程式也可以經由上節所介紹的一連串指令來找到其使用插座的編號然後再由其插座來使用 I ／O及 RAM scratchpad locations。當此副程式執行完成後，它能經由執行RTS（ Return from Subroutine)指令回到做這一個輸出的程式或語言去，它也可以跳到位址 $FDF0 的 COUT 副程式去執行將該字元顯示在螢光幕上然後再回到程式或語言去。

位置 $38 及 $39（十進位制的 56 及 57 ）稱之爲KSWL及KSWH或統稱爲鍵盤輸入開關（ Keyboard Input Switch 或簡稱爲KSW ）。這兩個位置內存放著使用中的 Apple 單字元輸入副程式之位址，通常此副程式（ 即 KEYIN ）的起始位址爲 $FD1B 。我們也可以經由 Monitor 的KEYBOARD 命令（ CTRL K ）及 BASIC 命令的 IN# 來改變這兩個位置內之內容爲 $Cn00 ，其中由這兩種命令給予插座編號 n 。每當 Apple 要由輸入設備來輸入一個字元時，Apple 即會依該位址呼叫插座編號 n 上的週邊面板之PROM的副程式，此時副

程式須將輸入字元送往6502微處理機的累積器上去並執行RTS指令而回到程式或語言中去。這個副程式須先設定輸入字元的高位元後再回去程式或語言中。

在週邊面板中的PROM內的副程式，它能夠改變CSW及KSW開關內的位址，而將此位址指示到其PROM中除了開始的位置之外的任何一個位置去。例如，某一PROM能夠在執行一段代碼後決定其所在的插座並且做好準備工作，然後再跳到實際在操縱字元的副程式去，而在此PROM做準備工作時，它能先將CSW及KSW開關改為此副程式的起始位址（到底是改變CSW或KSW開關應看是做輸出或輸入工作而定）。接著當Apple要求自該週邊面板輸入或輸出字元時，這一個操縱的副程式即會跳出已做完的準備工作而直接去執行應做的輸入或輸出工作。在對速度敏感的情況下這樣做可以節省一些時間。

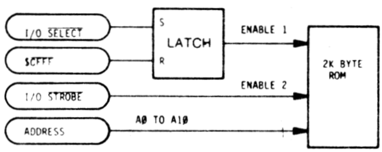
若PROM中有分開的副程式並各具有不同的功能而可依此來分別改變CSW及KSW開關，則我們即可利用一片週邊面板來同時執行輸入及輸出工作。在週邊面板上PROM之準備工作的步驟中，它可以經由檢視CSW及KSW的高位址部份來決定是被呼叫去執行輸入或輸出工作。不管那一個開關，若其位置內包含有位址$Cn則會被呼叫，被呼叫的即須去執行其工作，若是兩個開關內均包含位址$Cn時，那麼我們的副程式應假設是被呼叫去執行輸出工作。

八、擴充唯讀記憶體

記憶區配置中自位置$C800到CFFF止這2K的記憶體是保留給週邊面板中的2K ROM或PROM的，這2K的ROM或PROM是用來儲存龐大的程式或操縱副程式的。所謂擴充唯讀記憶體（expansion ROM）係利用Apple的記憶區配置中的位置，此種空間的利用可以使我們更便利的來寫介面程式（interface programs）。

這些擴充的PROM空間可以供給所有的週邊面板利用。另外也可以讓Apple上的一個以上的週邊面板來使用擴充的 ROM，只是每次只能供給一個擴充的ROM被使用。

每一個週邊面板的擴充的ROM均需一個起伏電路（ flip-flop ）來配合才有效，而這個起伏電路須經由DEVICE SELECT信號來轉到“on”狀態，這也就是說當我們在第一次參用開啟的週邊面板後每一面板上的expansion ROM均可以部份被啟用。另外在每一個週邊連接器上第20釘腳的I／O STROBE路綫也可用來使expansion ROM被啟用，每當Apple微處理機參用到expansion ROM範圍內的位置時均會使得此路綫開始動作，而當此路綫開始動作並且上面提到的起伏電路已處於“on”時，則Apple參用此一特別面板上的 expansion ROM。（見圖八）



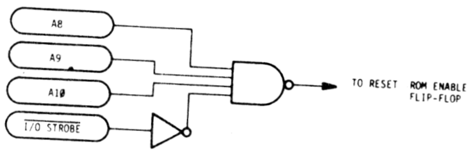
圖八　Expansion ROM Enable Circuit

一個週邊面板的256個位元組PROM能夠經由其做準備工作的副程式去參照位置$CFFF而獲得處理這些expansion ROM空間。$CFFF是一個特別的位置，所有的週邊面板均應將其視為一個可以將起伏電路“off”及停用 expansion ROM的信號。當然，此時也使得週邊面板無法自其面板上的expansion ROM取得其expansion ROM空間，但是當微處理機由256個位元組活動的PROM中取得另一個指令時即可使expansion ROM重新被使用。當expansion ROM

可以重新被使用時其內容已被清除乾淨，正在執行的副程式此時能跳到ROM中的程式去執行，它能增加2K的記憶體空間。

```
0332-   2C FF CF   BIT  $CFFF
0335-   4C 00 C8   JMP  $C800
```

經由不完全特殊位址$CFFF予以解碼而節省週邊面板上的電路（當然要犧牲 ROM空間）是可行的，事實上我們是可以損失 ROM空間中的最後256個位元組。下列的簡單電路是不錯的：



圖九　$CFXX Decoding

第六章　硬體的外形結構

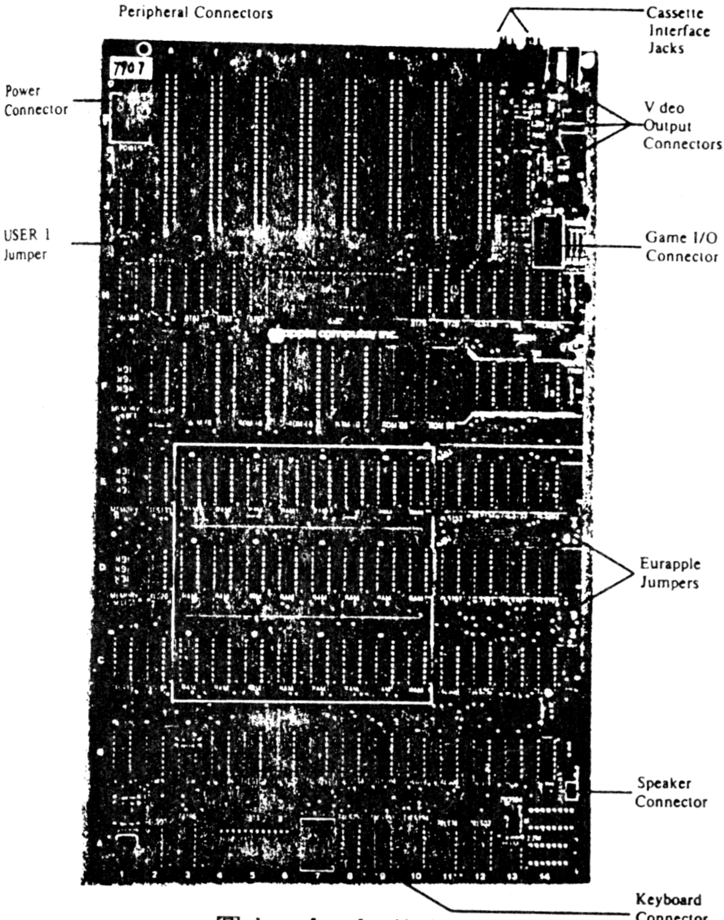
一、微處理機簡介（ THE MICROPROCESSOR ）

The 65Ø2 Microprocessor

Model:	MCS65Ø2/SY65Ø2
Manufactured by:	MOS Technology, Inc. Synertek Rockwell
Number of instructions:	56
Addressing modes:	13
Accumulators:	1 (A)
Index registers:	2 (X,Y)
Other registers:	Stack pointer (S) Processor status (P)
Stack:	256 bytes, fixed
Status flags:	N (sign) C (carry) V (overflow)
Other flags:	I (Interrupt disable) D (Decimal arithmetic) B (Break)
Interrupts:	2 (IRQ, NMI)
Resets:	1 (RES)
Addressing range:	2^{16} (64K) locations
Address bus:	16 bits, parallel
Data bus:	8 bits, parallel Bidirectional
Voltages:	+5 volts
Power dissipation:	.25 watt
Clock frequency:	1.023MHz

微處理機由下述的時序電路圖中獲得它主要的時序信號Φ0和Φ1，它們是 1,023 HZ 的時序信號。在不同的手册上包括MOS出的硬體技術手册，使用Φ2設計作爲Apple 的Φ0時序。

只有在時序Φ0作用的這一段時間週期裡，微處理機才使用它的位址和資料輸送共用綫(data bus)。當Φ0降低時，微處理機是在做它內部的操作，因此不需要位址和資料的輸送共用綫。



圖十 Apple的主電路板

這一微處理機有一個16位元的位址傳輸專綫（address bus），和一個8位元的前向資料傳輸專綫，這個位址傳輸專綫綫路是由在面板位置H 3，H 4及H 5的三個8T97產生三種值的緩衝器所滙集而成。這個綫路只有在DMA的廻路時才會保持開放的情況，在其他的時間內都是一直在動作著。位址值在位址傳輸綫上當Φ1升高時只有保持300×10^{-9}秒時間的正確性，但是在Φ0的週期內都永遠保持正確。

至於資料傳輸綫，就由二個8T28產生三值的緩衝器所滙集而成，這二個緩衝器是在面板上的H10及H11位置。從微處理機來的資料，放到傳輸綫上大約在Φ1的週期內有300×10^{-9}秒的時間，同時READ/WRITE的信號即（R／$\overline{\text{W}}$）同時降到0。在其他的所有時間裡，微處理機不是在注意資料傳輸綫就是根本不管它。

與微處理機相連的RDY，$\overline{\text{RES}}$，$\overline{\text{IRQ}}$，和$\overline{\text{NMI}}$這幾條綫路都使用3.3K歐姆的電阻使其保持高達十5伏的電壓。

另外與微處理機相連的SET OVERFLOW（SO），則固定不變的接地。

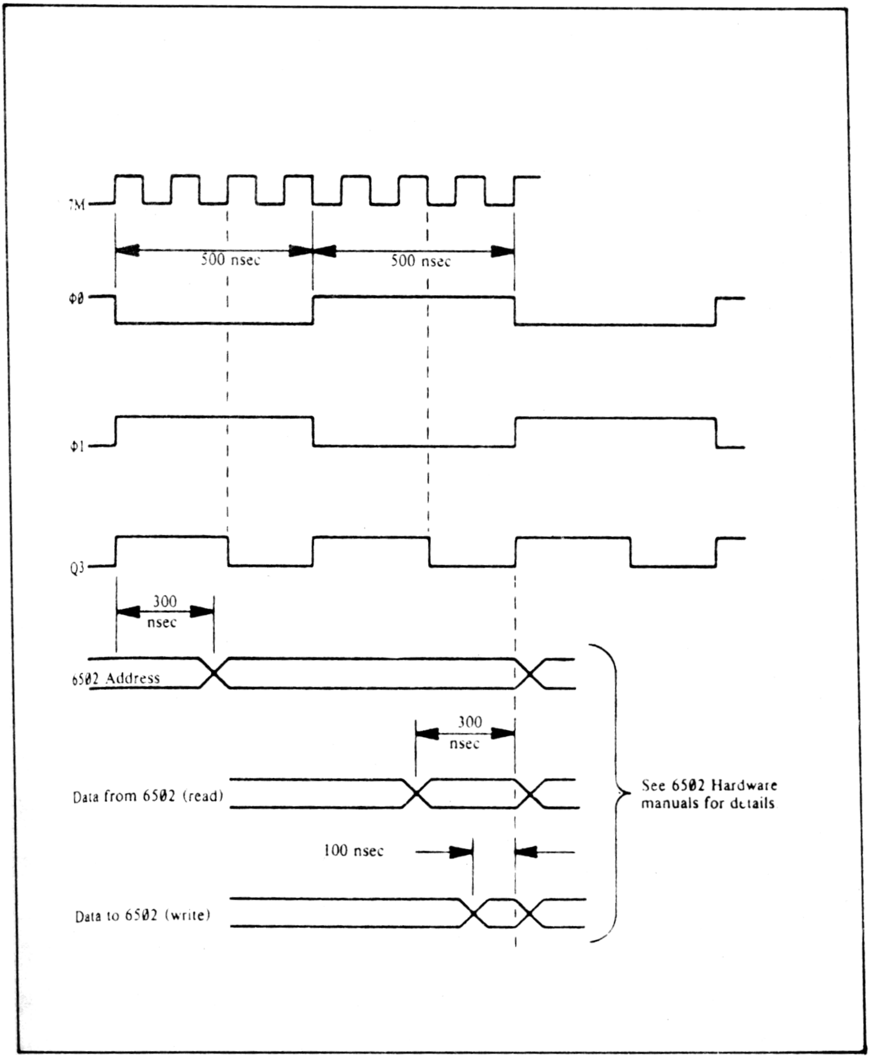
二、系統時序（SYSTEM TIMING）

Table 27: Timing Signal Descriptions

14M:	Master Oscillator output, 14.318 MHz. All timing signals are derived from this signal.
7M:	Intermediate timing signal, 7.159 MHz.
COLOR REF:	Color reference frequency, 3.580MHz. Used by the video generation circuitry.
Φ0 (Φ2):	Phase 0 system clock, 1.023MHz, compliment to Φ1.
Φ1:	Phase 1 system clock, 1.023 MHz, compliment to Φ0.
Q3:	A general-purpose timing signal, twice the frequency of the system clocks, but asymmetrical.

表二十七

所有的週邊設備連接器都能夠獲得時序信號7M，Φ0，Φ1，及Q3。而時序信號14M，及COLOR REF是週邊設備連接器所沒有辦法去獲得的。



圖十一　時序信號及其關係

三、電源供應器（POWER SUPPLY）

The Apple Power Supply (U. S. Patent #4,130,862)

Input voltage:	107 VAC to 132 VAC, or 214 VAC to 264 VAC (switch selectable*)
Supply voltages:	+5.0 +11.8 −12.0 −5.2
Power Consumption:	60 watts max. (full load) 79 watts max. (intermittent**)
Full load power output:	+5v: 2.5 amp −5v: 250ma +12v: 1.5 amp (~ 2.5 amp intermittent**) −12v: 250ma
Operating temperature:	55c (131° Farenheit)

Apple 的電源供應器是一種高伏特的開關式電源供應器。在大部分的別種電源供應器上都是使用許多繞綫式的轉換器來改變輸入的電壓，使其降低許多伏特然後再整流及調節這些降低的電壓，而Apple的電源供應器却是首先把交流電壓轉換成直流電流，然後利用這直流電壓去驅動一個高頻率的振盪器，這個振盪器的輸出再接入一個小的綫繞式的轉換器，然而在第二層繞綫的電壓往往被調整爲Apple的輸出電壓。

＋5伏特的輸出電壓是與一參用的電壓比較，同時這不同的誤差被傳回到振盪器的綫路裡，當 Apple 的電源供應器的輸出開始超過它的負載量時，振盪器的頻率將隨著改變，這樣電壓就會回復到它的正常標準。

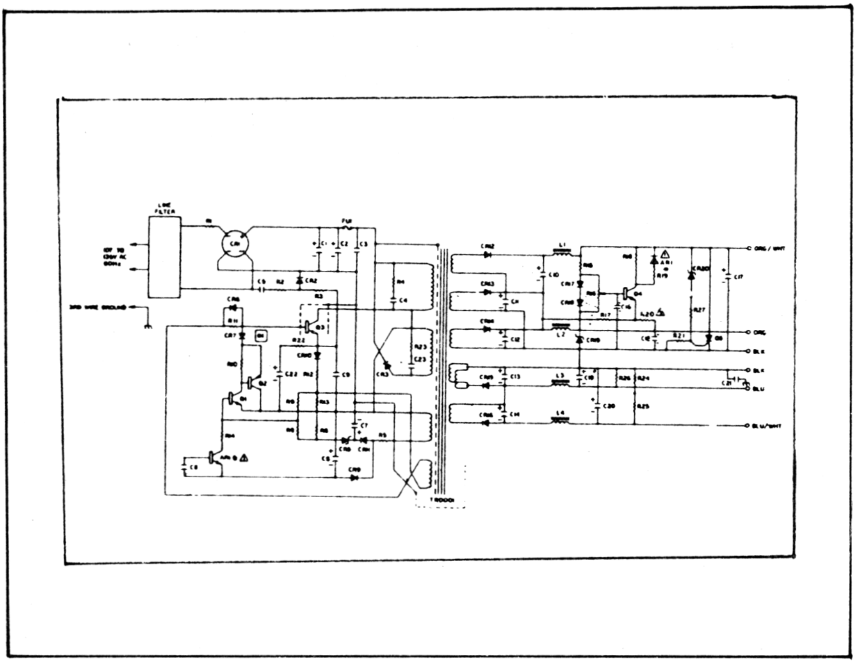
假如說突然在電源供應器的一個輸出電壓產生綫路短的情況時，在 Apple 電源供應器裡的一個傳回的綫路就會停止振盪器的振動同時中斷所有的輸出電路。這時電源供應器便會暫停大約0.5秒，然後再企圖去重新開始振動。如果輸出還是在短路中，振盪器將再停止同時再等待，它會不斷做這樣的循環，直到短路的綫路被拿開或是電源被關掉。

如果電源供應器的輸出連接器沒有接到 Apple 的主要面板上，電源供應器將會注意到這一個沒負載的情況同時會有效的自動短路，像這種保護綫路及中斷所有的電源輸出已在前面提到，這樣做可以避免電源供應器內部損害的危險。

如果由於電源供應器的內部或是外在的任何問題，在輸出的任一電壓，脫離了它可負載的階段，它便會再一次的自己關閉掉以避免損害在主要面板上的其它零件。這樣可以確定所有電壓將不是調整正常及相稱就是被停止供應。

當上面所提到一些失常的情況發生時，內部的保護綫路就會停止驅動轉換器的振動。經過一個短暫的時間以後，電源供應器將執行一個重新起動的廻路，同時希望能再振動。如果失常的情況沒有被解除掉，供應器將再一次的關閉。這樣的廻路能無限制的繼續做而不會損壞到電源供應器，每一次振盪器關閉和重新啟動它的頻率產生可聽到聲音，所以你可以聽到電源供應器產生一種吱吱的尖銳聲。因此只要失常的情況發生，你就能聽到從電源供應器裡發出的“喀喀喀”的響聲，這就是警告你在電源的輸出上有了錯誤。

無論如何不要供應大於140VAC到轉換器的輸入位置（如果供應器的開關是在220V的位置則不可大於280VAC），否則供應器將會永遠的壞掉。所以說無論如何你必須在Apple的電源供應器外面做一接地線，這是一件很重要的事。



圖十二　電源供應器的電路圖

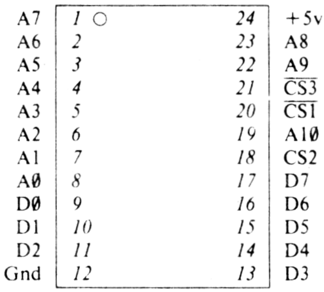
四、唯讀記憶體（ROM　MEMORY）

Apple能提供至多6個2K的唯讀記憶體的IC，而這些IC裡可容納8個內在的程式當任一時期間微處理的位址傳輸綫存有在$D000與$FFFF之間的位址時，在Apple的面板上位置F12的一個74LS138則會啟動6個唯讀記憶體的一個。在所有ROM裡的八個資料的輸出都是與微處理機的資料線暫存器相連接，同時僅讀記憶體的位址綫被接到驅動微處理機的位址綫A0到A10的暫存器。

唯讀記憶體有三個零碎選擇綫使這記憶體能作用。CS1和CS3兩者都是與74LS138連接在一起，連接位置在F12，且可選擇個別的僅讀記憶體，CS2則對所有的ROMs都可共同使用，同時連接到週

邊連接器的 $\overline{\mathrm{I\ N\ H}}$ 線。如果在週邊插座內的任一週邊面板，把這一線路拉到“低”值，則所有在Apple主要面板上的ROM皆停止活動。

這些ROM與形式2316和2716的PROM相似，但是在大部份的這些PROM裡晶片選擇線（chip selects）與ROM有不同的極性，所以PROM不能直接的被接到Apple的主要面板上去。



圖十三　9316B ROM Pinout

五、隨機取存記憶體（RAM MEMORY）

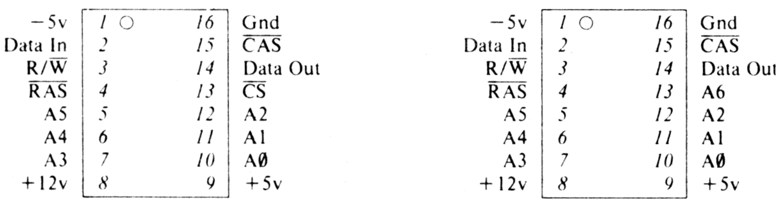
Apple使用4K及16K的動態式隨機取存的記憶體爲它主要的隨機取存儲存區。這些RAM記憶體同時都被微處理機和螢光幕顯示電路所使用。微處理機和螢光幕顯示器交互使用這些RAM：微處理機在Φ0時讀取或寫入資料到RAM，而顯示器在Φ1時把RAM記憶體的資料搬到螢光幕上。

在位置E11，E12和E13的三個74LS153，及在位置E14的74LS238，及在C12的74LS257的半個組織成RAM記憶體的複合位址器。這位址器從微處理機所產生的位址及螢光幕產生器所產生的位址複合起來，放到6個RAM的位址線。其它的RAM位址信號，$\overline{RAS}$ 和 $\overline{CAS}$ 以及16K RAM的位址線第6線及4KRAM的 $\overline{CS}$ 都

是由RAM選擇綫路所產生。這一選擇綫路及2個在位置E2及F2的74LS139及在位置C1的74LS153的一半及在位置C12的74LS257及在J1的74LS257的一半所組合而成，同時這三個記憶體的形態聚集在位置D1，E1及F1這一綫路訂定固定的信號綫路到每列的RAM，依照在這一列裡RAM的形式（4K或16K）爲依據。

這動態式的RAM在整個Φ1時序態自動的被螢光幕產生器的電路所復還，因爲螢光幕顯示幕最少顯示1K的記憶體，它對在這一1K的記憶的每個位置，必須在一秒鐘內循迴60次。也因爲如此，這一動作能自動的清洗在這48K值元組的RAM裡的每一位元。這樣與顯示器和微處理機的交互處理廻路相互連接，可使螢光幕顯示器，微處理機，和所有的RAM以很高的速成清除掉，而互相之間不會相妨礙。

資料輸入到RAM裡直接脫離系統的資料共用傳輸綫，資料從RAM輸出則由在面板位置B5和B8的二個74LS174所閂住，同時與Apple的鍵盤的7位元的資料複合，這些被閂住的RAM輸出被直接輸入到螢光幕顯示器的文字，彩色，和點產生器，同時由二個在面板位置B6和B7的74LS257轉回系統的資料傳輸共用綫。



4096 4K RAM
Pinout

4116 16K RAM
Pinout

圖十四　RAM　Pinouts

六、螢光幕產生器（THE VIDEO GENERATOR）

在螢光幕上有192條掃射線，分成8組每組掃射線有24條，每一掃射線顯示部份或是所有40位元組儲存體的內容。

螢光幕產生器電路從在面板位置D11到D14結合的74LS161計數器驅動它的同步及時序信號。這些計數器產生15個同步信號：

H0 H1 H2 H3 H4 H5

V0 V1 V2 V3 V4

VA VB VC

信號有“H”表示這一組就在螢光幕上的水平位元組，這些位元組從000000 到100111 二進位表示（十進位39）。信號V0到V4是在螢光幕上的垂直線位置，二進位表示從00000到10111（十進位23）。VA，VB，VC信號是在螢光幕上的垂直線內的垂直掃射線的位置，二進位表示從000到111（十進位7）。

這些信號被送到RAM位址複合器，位址複合器再把這些信號轉換成單一位址的RAM位置，轉換依據螢光幕顯示器形態軟體開關的設定（下面會提到）。在時序Φ1升高時，RAM的位址複合器就把位址送到RAM記憶體的行列位置。保有從RAM行列送來的RAM資料的插梢，便重新固定路線到螢光幕產生器的電路。在位置上的74LS283重新排定記憶體的位址所以在顯示幕的的記憶區分配圖就被改變。

如果在顯示幕上目前的區域是一個編輯字元，這時螢光幕產生器將把資料的6個低位元固定線路到位置A5上的2513型式之字元產生器。在每一個字元的7個到顯示點由VA，VB，VC信號所掃描出來，同時字元產生器的輸出被在位置A3的一個74166排列成一行點串。這位元串被固定到一個Exclusive-OR邏輯器，如果資料位元組的

高位元是“ off ”值同時第六個位元是“ low ”值或在位置B3的555計時器是“ high ””值，這一邏輯值就會被轉換。這樣會產生一個相反和閃動的字元，這一編輯位元串這時便被送到螢光幕選擇複合器。

如果Apple的螢光顯示幕是在繪圖的形式，這樣從RAM來的資料被送到在面板位置B4和B9的二個74LS194位移暫存器。在這裡每一圖形都轉成線性的資料串。這二個資料串都同樣被傳送到螢光幕選擇／複合器。

在電路板上位置A8的74LS257組合器能從彩色和高解析度的圖形顯示中選擇一個，如果位元組上的高位元是設定爲1，則成綫形的高密度點串被在位置A11的74LS74延遲半個時序廻路，這樣可產生在高解析度的繪圖形態的改變色彩組。

這個螢光幕選擇／複合器混合上面所提到二種資料串的來源，由於把螢光幕顯示幕的軟體開關設定值的關係。在位置A10的74LS194和在位置A9的74LS151從彩色繪圖，或高解析度的繪圖選擇一個綫形位元串，就是依據螢光幕的形態。最後的綫形輸出是複合混合同步信號及由螢光幕同步綫路產生的充滿彩色的信號，然後送到螢光幕輸出連接器。

控制螢光幕形態的螢光幕顯示軟體開關，被解碼後做爲Apple電路板上的I／O功能的一部份。在電路板上的位置B12，B13，B11，A12，和A11 的邏輯閘都被用來控制不同的螢光幕形態。

充滿色彩的信號足由B12，B13，及C13 的邏輯閘所產生，同時由R5，coil L1，C2，及整容器C3 所支配。這個 C3 整容器能被用來轉成改變由螢光幕顯示器所產生的色彩的濃度。當Apple是在顯示編輯的形式下時，電晶體Q6和與它成對的電阻R27，可使充滿色彩的信號停止活動。

七、螢光幕的輸出插孔（VIDEO OUTPUT JACKS）

由上面所提到的電路所產生的螢光幕信號，是一種與NTSC可共處且與EIA相似的標準相互混合成的螢光訊號。而這種信號能被輸入任何標準封閉電路成攝影螢光幕監督器，這一種信號可在Apple電路板上的下列三個地方獲得：

RCA插孔：在Apple電路板的後面靠近右邊的地方就是標準RCA插孔。這一插孔的套管被接到Apple共同的地綫，這插孔的頂端被接到經由一個200歐姆電位計的螢光幕輸出信號，這個電位計能調節連接器的電壓從0到1V。

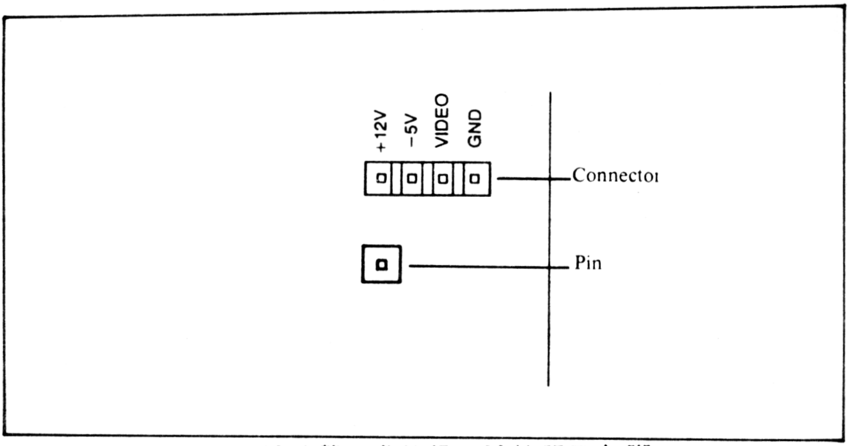
輔助螢光幕連接器：在Apple電路板的右邊靠近背面是一個Molex KK 100 線形連接器，備有四個方形釘脚，0.25″高，及0.01″的中心點。這一連接器供應混合成的螢光幕輸出及二個電源供應器的電壓，這一連接器圖示於圖15。

輔助螢光幕釘脚：這單一的綫包的金屬釘脚，在輔助螢光幕輸出連接器的下方，可供應與上面連接器相同的螢光幕信號。這一釘脚主要是爲歐洲型Apple PAL/SECAM改裝電路板上的一個連接點。

Table 28: Auxiliary Video Output Connector Signal Descriptions

Pin	Name	Description
1	GROUND	System common ground; 0 volts.
2	VIDEO	NTSC compatible positive composite video. Black level is about .75 volt, white level about 2.0 volt, sync tip level is 0 volts. Output level is not adjustable. This is not protected against short circuits.
3	+12v	+12 volt power supply.
4	−5v	−5 volt line from power supply.

表二十八



圖十五　輔助螢光幕的輸出連接器及釘脚

八、固定的輸入與輸出（BUILD－IN I／O）

Apple的固定的輸入與輸出功能分配於128個記憶位置，由＄C000開始，在Apple 的電路板上在位置F13的一個74LS138 稱爲輸入與輸出選擇器解碼，這128個特殊位址的內容能同時起動這些不同的功能。

無論任何時候Apple的位址傳輸線上存有＄C000到＄COFF間之任一位址時，則在位址H12上的74LS138會啟動在位置F13的74LS138。這個輸出與輸入選擇器把這一256位元組的排列分割成8個16位元組的排列而不管＄C080到＄COFF的排列。在74LS138裡的每一條輸出綫在與它有相同關係的16位元組的排列被參用到時這這一輸出綫便開始活動（在低時序）。

在輸出與輸入（以下簡稱I／O）的選擇器中標爲“0”的綫路，可搜集從鍵盤連接器的資料，並存於RAM資料組合器。

在I／O選擇器標爲“1”的綫路，可重新設定在位置B10到74LS74交互續閘，而這一交互接續閘就是鍵盤的旗標（flag）。

在I／O選擇器上標爲“ 2 ”的綫路，套住半個在位置K13的74LS74 。這一交互接續閘的輸出被接到經由一個電阻網路到卡式磁帶輸出插孔的頂端。

在I／O選擇器上標爲“ 3 ”的綫路，套住另一半在位置K13的74LS74。這一交互接續閘的輸出被接到經由一個需容器和達林頓安培計綫路到Apple的發音連接器，這一發音連接器在電路板的右邊緣之鍵盤的下方。

在I／O選擇器上標爲“ 4 ”的綫路是直接被接到Game I／O連接器的5號釘脚，這一支脚就是通用程式$\overline{\text{C040 STROBE}}$ 。

標爲“ 5 ”的綫路被用來做爲啟動在位置F14上的74LS259 ，這一個 IC 包含做爲螢光幕顯示的軟體開關以及Game I／O連接器操動輸出。這些開關是由位址綫路1到位址綫路3來選擇，同時每一開關的設定是由位址綫0來控制。

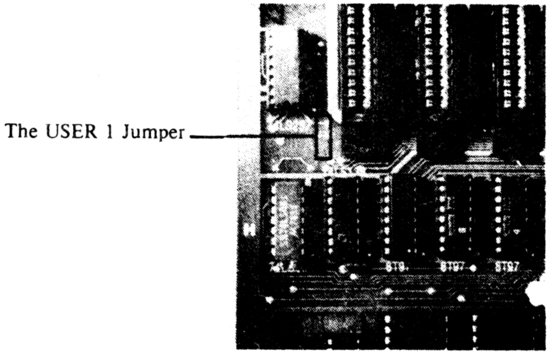
標爲“ 6 ”的綫路是被用來啟動在位置H14的一個8位元的組合器74LS251，當這一組合器被啟動時，則速接其8個輸入綫路中的一條到有三情況的系統資料傳輸綫的高位元，（即第七個位元）。在底邊的三條位址則控制組合器所選擇的8個輸入中的一個。組合器中的四條輸入綫是由在位置H13上的一個四方形計時器接過來的。至於輸入到這一計時器的，則是在GAME I／O 連接器上的GAME控制釘脚。組合器的另外三條輸入綫取是由Game I／O連接器上的單一位元輸入接過來。組合器上的最後一條輸入綫，則是由在位置K13上的一個741用於操作的安培計上接過來。至於輸入到這一安培計的綫路，則是由卡式磁帶輸入插孔而來。

I／O選擇器上標號“ 7 ”的綫路，則是用於重新設定在位置H13 上的553四方形時序產生器內的所有四個時序產生器。輸入到這時序器的四條輸入綫，則是由一個RC網路接過來。這一RC網路是

由4個0.022μF電容器，4個100歐姆的電阻器，及在GAME控制器上接觸到Game I／O連接器上的不同的電阻器。在四個時序電路內的電阻都有不同的總和阻力，而也因這些阻力的不同決定了這一電路的時序特性。

九、“USER 1”JUMPER

在Apple的電路板上有一對無法標示的銲接的墊套，它在0號插座的左下方，這一對墊套稱爲“User 1”jumper。下列圖示這一jumper的位置，如果你在這二個墊套之間連接上一條電綫，這樣在每一週邊連接器上的“User 1”綫則可使用。如果任何週邊面板，把這一綫路拉到低位值，則所有內部的I／O解碼工作將無法執行。I／O SELECT和DEVICE SELECT綫路，在USER 1是低位值時，都將變爲高位值，同時一直保持高位值，而不管在位址傳輸共用綫上的位址值。



照片八 The USER 1 Jumper

十、遊戲的輸入與輸出連接器（THE GAME I/O CONNECTOR）

Signal	Pin	Pin	Signal
+5v	1 ○	16	NC
PBØ	2	15	ANØ
PB1	3	14	AN1
PB2	4	13	AN2
$\overline{\text{CØ4Ø}}$ $\overline{\text{STROBE}}$	5	12	AN3
GCØ	6	11	GC3
GC2	7	10	GC1
Gnd	8	9	NC

圖十六 Game I/O Connector Pinoute

Table 29: Game I/O Connector Signal Descriptions

Pin:	Name:	Description:
1	+5v	+5 volt power supply. Total current drain on this pin must be less than 100mA.
2-4	PBØ-PB2	Single-bit (Pushbutton) inputs. These are standard 74LS series TTL inputs.
5	$\overline{\text{CØ4Ø}}$ $\overline{\text{STROBE}}$	A general-purpose strobe. This line, normally high, goes low during ΦØ of a read or write cycle to any address from $CØ4Ø through $CØ4F. This is a standard 74LS TTL output.
6,7,10,11	GCØ-GC3	Game controller inputs. These should each be connected through a 150K Ohm variable resistor to +5v.
8	Gnd	System electrical ground.
12-15	ANØ-AN3	Annunciator outputs. These are standard 74LS series TTL outputs and must be buffered if used to drive other than TTL inputs.
9,16	NC	No internal connection.

表二十九

十一、鍵盤（THE KEYBOARD）

Apple之固有的鍵盤是由一個整體的鍵盤解碼ROM之MM 5740而建立。在這一ROM上的接脚4到12以及22到31 的輸入綫是與在鍵盤電路上的鍵開關的矩陣相連接。這一個ROM的輸出被一個7404所緩衝，同時被連接到Apple的鍵盤電路連接器。

鍵盤電路解碼器很迅速的掃描過在鍵盤電路上的鍵行列，尋找是那一個鍵被按進來了。這一掃描的動作是由一個自由跑動的振盪器所控制，這一振盪器是由在鍵盤位置U 4的7400裡的三個段落所組成。這一振盪器的速度是被C6，R6及R7所控制，這幾個C6，R6，及R7是在鍵盤的印刷綫路電路上。

在鍵盤上的 REPT 鍵與一個555時序產生器電路相接，這一時序產生器電路是在鍵盤電路板的U 3位置。這一脚座和這電容器及週圍的3個電阻器產生這10Hz"REPeaT"信號，如果把220K歐姆的電阻換成另一個較低値的電阻，那樣這個 REPT 鍵將以較快的速度來重複這一個輸入字元。圖十七卽爲Apple的鍵盤電路。

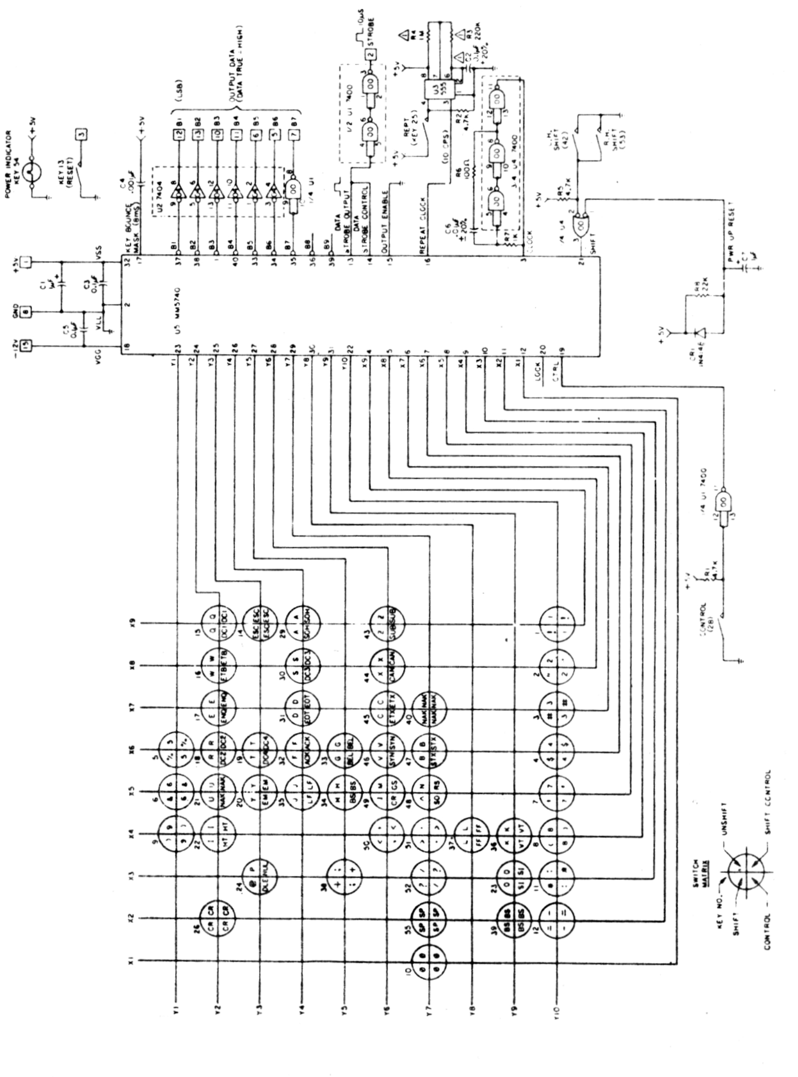


Figure 17. Schematic of the Apple Keyboard

圖十七　Apple 鍵盤的電路圖

十二、鍵盤的連接器（KEYBOARD CONNECTOR ）

從Apple的鍵盤電路來的資料直接送到RAM資料組合器，同時固定住在位置B6和B7的二個74LS257s。在鍵盤電路連接器上的STROBE綫，把在位置B10的74LS74交互接續開設定為1值。當I／O選擇器啟動它的“0”綫路時，在鍵盤電路連接器的七個輸入綫的資料及特殊指示信號的交互接續開的值，被組合到Apple的資料共用傳輸綫上。

Table 30: Keyboard Connector Signal Descriptions

Pin:	Name:	Description:
1	+5v	+5 volt power supply. Total current drain on this pin must be less than 120mA.
2	STROBE	Strobe output from keyboard. This line should be given a pulse at least 10μs long each time a key is pressed on the keyboard. The strobe can be of either polarity.
3	$\overline{\text{RESET}}$	Microprocessor's $\overline{\text{RESET}}$ line. Normally high, this line should be pulled low when the [RESET] button is pressed.
4,9,16	NC	No connection.
5-7, 10-13	Data	Seven bit ASCII keyboard data input.
8	Gnd	System electrical ground.
15	−12v	−12 volt power supply. Keyboard should draw less than 50mA.

表三十

+5v	1 ○	16	NC
STROBE	2	15	−12v
RESET	3	14	NC
NC	4	13	Data 1
Data 5	5	12	Data Ø
Data 4	6	11	Data 3
Data 6	7	10	Data 2
Gnd	8	9	NC

圖十八　Keyboard Connector Pinouts

十三、卡式磁帶機介面板的挿孔（CASSETTE INTERFACE JACKS）

在Apple II 電路板的背後有二個陰性小型的電話挿孔能夠連接你的Apple到一個一般的家庭用卡式磁帶錄音機。

卡式磁帶輸入挿孔：這一個挿孔是設計來作爲與大部份的磁帶錄音機上的耳機或是監督器的輸出挿孔相連接，這一輸入的電壓必須在，伏特尖峯之間，這一輸入的阻抗（ impedance ）是 12 K歐姆。

卡式磁帶輸出挿孔：這一挿孔是設計來作爲與大部份的磁帶錄音機上的麥克風輸入相連接，這一輸出的伏特是 25mv 到一個 100 歐姆的阻抗負載。

十四、電源連接器（POWER CONNECTOR ）

這一連接器與從Apple電源供應器接出來的電纜配成一對，這是一個AMR # 9－35028－1 六釘脚的陽性連接器。

Table 31: Power Connector Pin Descriptions

Pin:	Name:	Description:
1,2	Ground	Common electrical ground for Apple board.
3	+5v	+5.0 volts from power supply. An Apple with 48K of RAM and no peripherals draws ~1.5 amp from this supply.
4	+12v	+12.0 volts from power supply. An Apple with 48K of RAM and no peripherals draws ~400ma from this supply.
5	−12v	−12.0 volts from power supply. An Apple with 48K of RAM and no peripherals draws ~12.5ma from this supply.
6	−5v	−5.0 volts from power supply. An Apple with 48K of RAM and no peripherals draws ~0.0ma from this supply.

表三十一

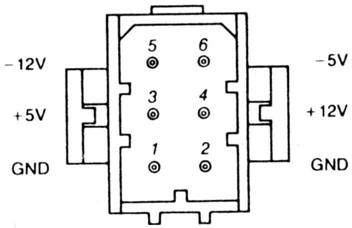


Figure 19. Power Connector

圖十九　電源連接器

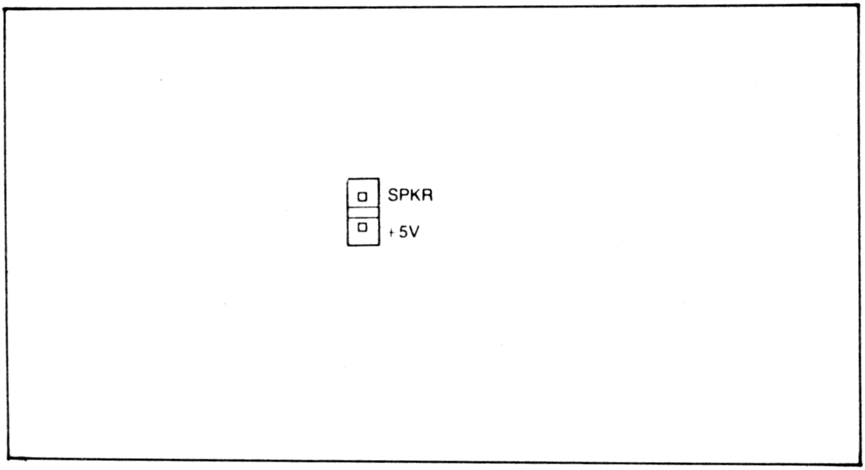
十五、發音器（ SPEAKER ）

Apple的固有的內部發音器，是由半個的74LS74交互接續閘並經由一個達林頓擴大器電路所帶動。這一個發音器的連接器是一個Molex KK100綫形連接器，其中包括二個方形的釘脚，而每個釘脚高0.025″及0.10″的中心點。

Table 32: Speaker Connector Signal Descriptions

Pin:	Name:	Description:
1	SPKR	Speaker signal. This line will deliver about .5 watt into an 8 Ohm load.
2	+5v	+5 volt power supply.

表三十二



圖二十　發音器的連接器

十六、週邊設備連接器（PERIPHERAL CONNECTORS）

沿著Apple電路板背面邊緣的8個週邊設備連接器是一個Winchester #2HW25C0－111的50釘脚PC板邊緣連接器，其每支釘脚爲0.10″的中心點。圖二十一列示此連接器的各釘脚名稱，表三十三，則爲這些信號的詳細描述。

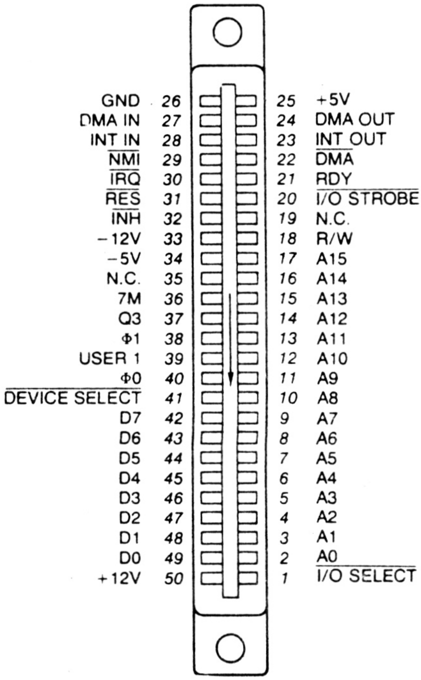


Figure 21. Peripheral Connector Pinout

圖二十一 **Peripheral Cennector Pinout**

表三十三　週邊設備連接器信號說明

Pin：	Name	說明：
1	$\overline{\text{I/O}}$ $\overline{\text{SELECT}}$	這一條綫在一般情況均處理方面，但當微處機參用到＄Cn 儲存頁時 則會變爲低值，n 表示每一個插座編號。這一信號在Φ0時序間變成活動，同時驅動10LSTTL 的負載。這一信號不會出現在週邊連接器0上。
2–17	AO–A15	這是被緩衝的位址傳輸共用綫，在這些綫上的位址在Φ1的時序間變成有效的位址，同時保持有效到Φ0。這些綫將有每條綫驅動 5 LSTTL負載的能力。
18	R/$\overline{\text{W}}$	被緩衝的讀與寫信號，這條綫與地址傳輸共用綫的有效的時間相同，同時在讀的廻圈時升爲高值，在寫的廻圈時降低值，這條綫能驅動達到2LSTTL的負載。
19	SYNC	這一釘脚只在週邊連接器7才有，這一釘脚被接到電視時序產生器上的SYNC信號上。
20	$\overline{\text{I/O}}$ $\overline{\text{STROBE}}$	當位址傳輸共用綫包含有一個在＄C800 到＄CFFF 之間的位址時這一條綫在Φ0間則爲低值。這一條綫可驅動 4 LSTTL的負載。
21	ROY	這是6502的RDY輸入，把這一條綫在Φ1時拉回低值，則會使微處理機暫時停止使用，同時在位址傳輸綫上擁有現在已被接到的位置的位址。
22	$\overline{\text{DMA}}$	把這一綫拉向低值，則使6502的位址傳輸共用綫將停止作用同時暫停微處理機的作用

		。這一條綫被－3 K歐姆的電阻器保持高值到＋5伏特。
23	INTOUT	如雛菊似相連的挿斷輸出到較低的優先順序排列的設備，這一釘脚通常與釘脚28相 連接。
24	OMAOUT	如雛菊似相連的DMA輸出到較低的優先順序排列的設備，這一釘脚通帶接到釘脚22（DMA IN）
25	＋5 V	＋5伏特的電源供應器，對所有的週邊面板而言500mA 電流是可獲得的。
26	GNO	系統電流的地綫。
27	DMA IN	從較高的優先順序排列設備傳入的雛菊似相連接的DMA較入通常與接脚24 相連接（DMA OUT）
28	INT IN	從較高的優先順序設備傳入的如雛菊似的相接連的挿斷輸入通常與釘脚23 相接。
29	$\overline{NMI}$	當這一條綫被拉向低值時，Appl e開始一個挿斷廻路，同時跳到挿斷斷操作副程式，這一副程式在位置＄3FB。
30	$\overline{IRQ}$	挿斷要求。當這一條綫被拉回低值時並且在6520的I旗標（挿斷停止旗標）沒有被設定值時才能執行挿斷廻路。如果是這樣，6520將跳到挿斷操作副程式去執行，這一挿斷副程式位址存於位置＄3FE到＄3FF。
31	$\overline{RES}$	當這條綫被推向低值，則這微處理機開始一重設（RESET）廻路。

32	INH	當這條線被推向低值，在Apple電路板上的所有ROMs 都將無法活動。這條線被－3 K歐姆的電阻保持高值到＋5 V。
33	－12V	-12伏特的電源供應器，對所有的週邊電路板而言，最大電流是200mA。
34	－5 V	－5伏特的電源供應器，對所有的週邊電路板而言，最大電流是200mA。
35	COLOR REF	這條線，只在週邊連接器7上才有，這一釘脚連接到螢光幕產生器上的3.3MHzCOLOR參用信號。
36	7 M	7MH2時鐘序，這條線將驅動2LSTTL負載。
37	Q 3	2MH2不對稱的時鐘序，這條線將帶動2LSTTL負載。
38	Φ 1	微處理機的現象，表示one clock，這條線將驅動2LSTTL負載。
39	USER 1	當這條線被推向低值，使所有的內部 I／O位址解碼動作皆無法活動。
40	Φ 1	微處理的零時鐘序的現象，這條線將驅動2LSTTL負載。
41	DEVICE SELECT	當位址傳輸共用線持有一位址是在$COnO到$COnF時，在每一週邊連接器上的這一條線則變成可活動（即在低值），上面$COn0及$COnF中的n表插座編號加$8，這條線將驅動10LSTTL負載。
42－49	D0－D7	緩衝的向前資料傳輸共用線。在這條線上的資料在寫的廻路Φ0有 300×10^{-9}秒的有效

		時間，同時將固定著，在讀的廻路Φ 0 結束以前，不會少於 100×10^{-9}秒，每一資料綫能驅動一個 LSTTL 負載。
50	＋12 V	＋12 伏特電源供應器，這個能供擴最高到 250mA 的總和電流。這一電流對所有的週邊面板都一樣。

圖二十二　Apple II 的電路圖

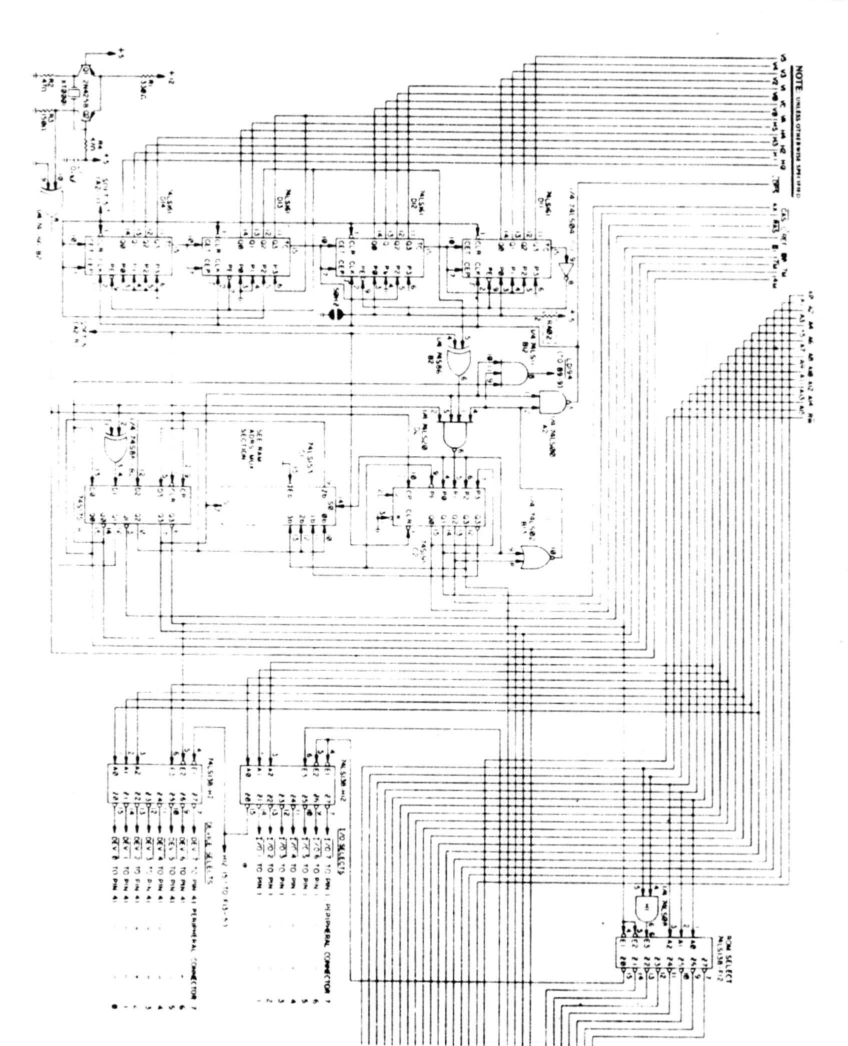


Figure 22-1. Schematic Diagram of the Apple II

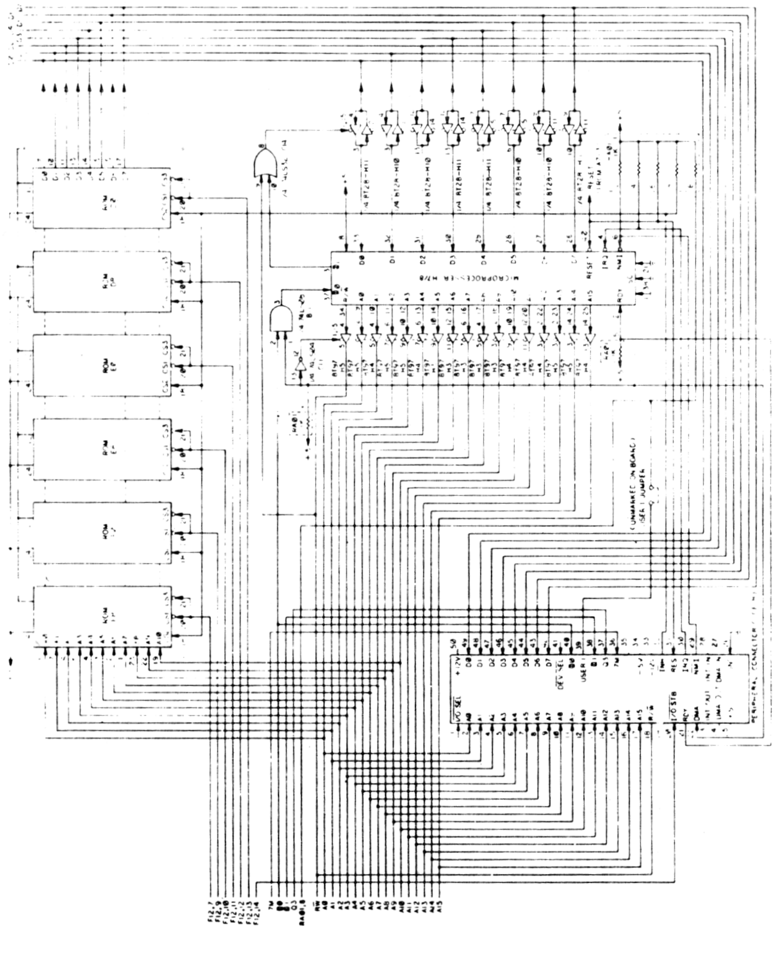


Figure 22-2. Schematic Diagram of the Apple II

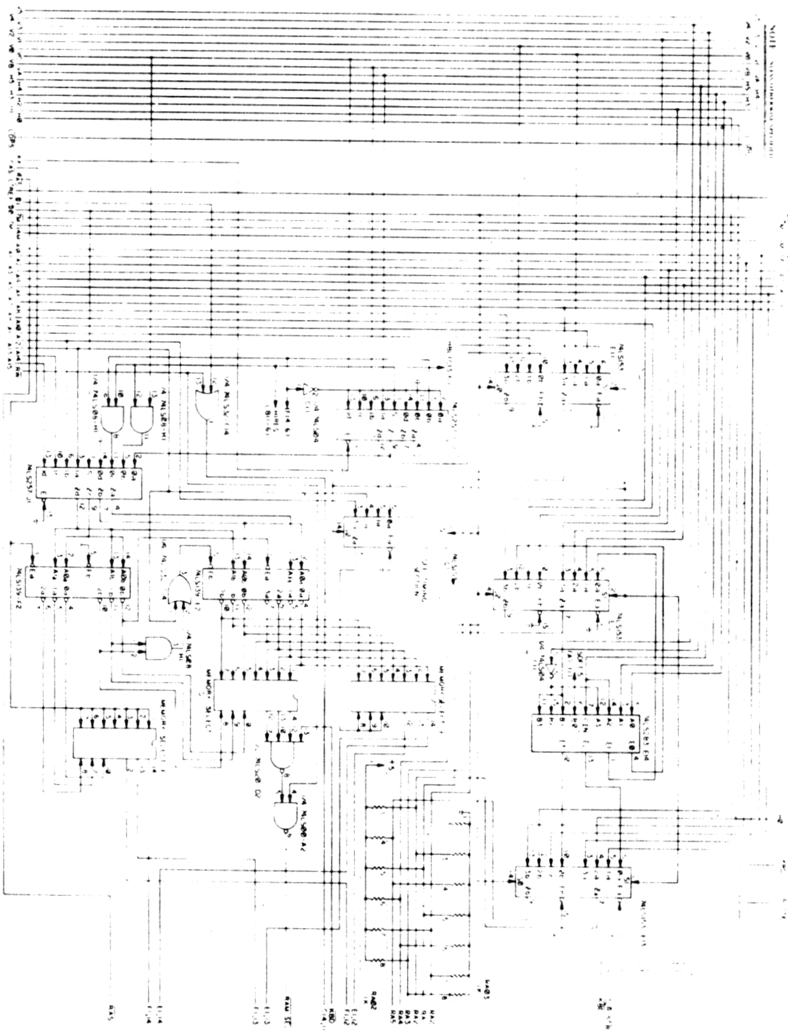


Figure 22-3. Schematic Diagram of the Apple II

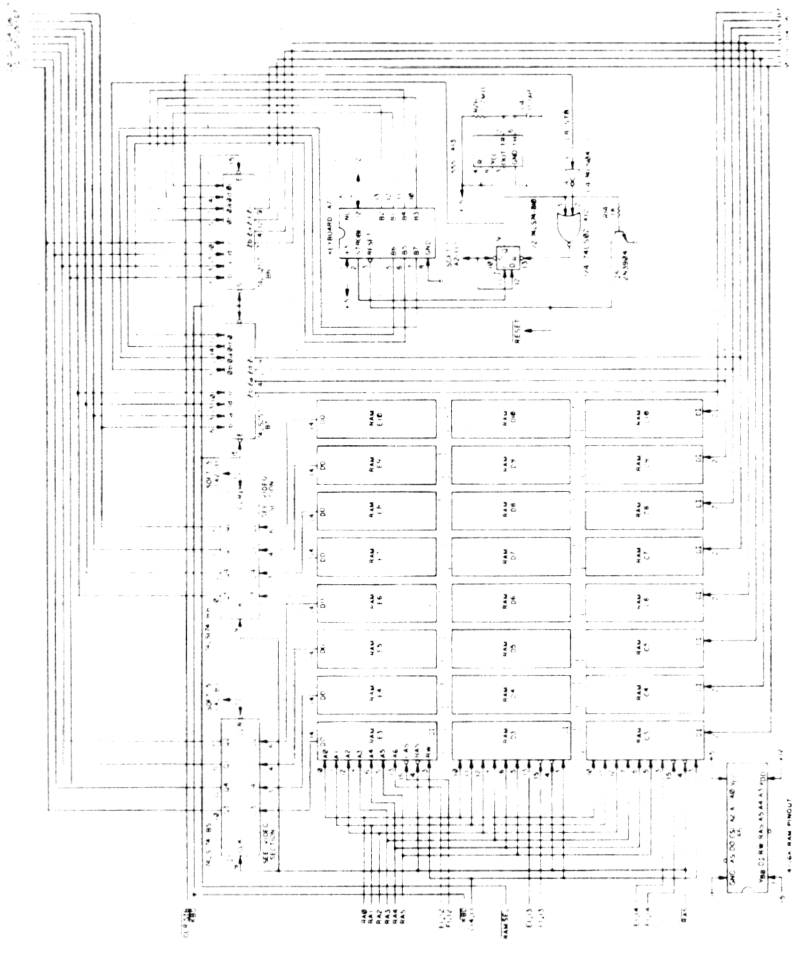


Figure 22-4. Schematic Diagram of the Apple II

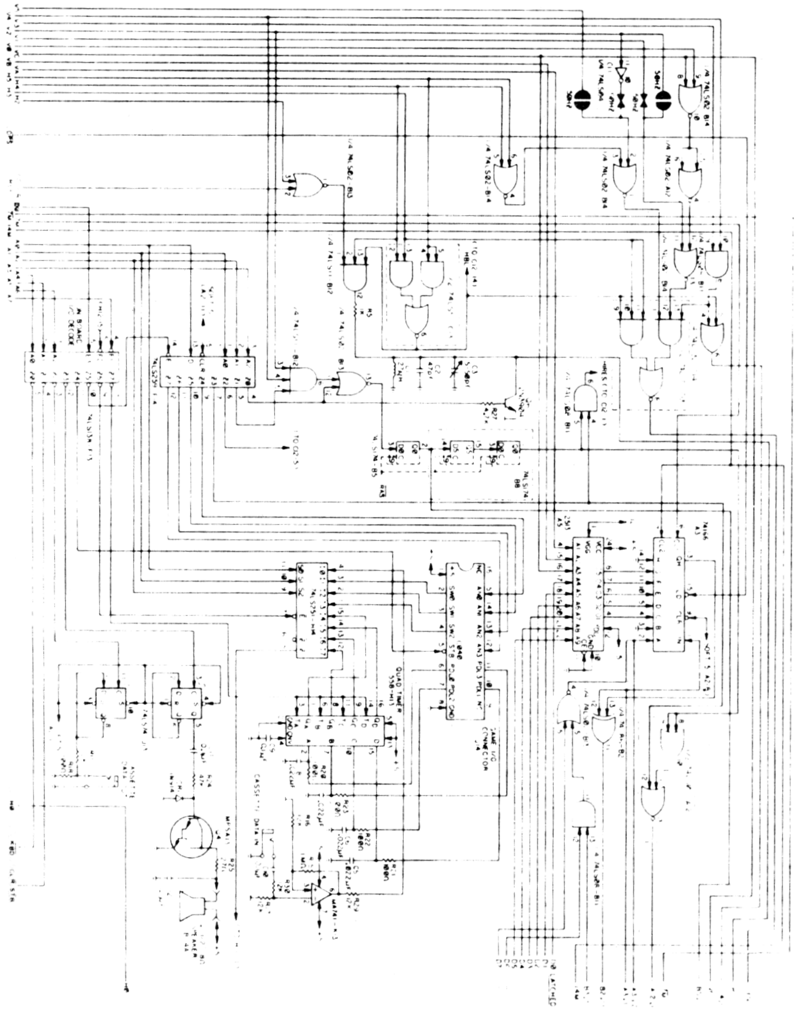


Figure 22-5. Schematic Diagram of the Apple II

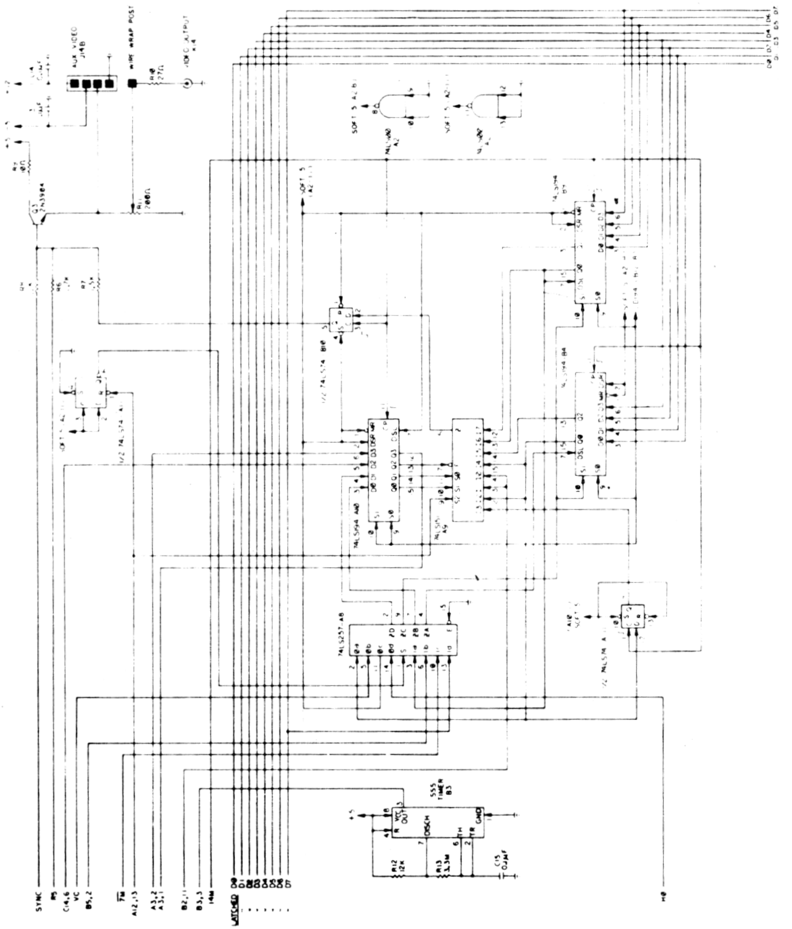


Figure 22-6. Schematic Diagram of the Apple II

附　　錄

附錄A　THE 6502 INSTRUCTIONS SET ·············· 158

附錄B　SPECIAL LOCATIONS ························· 170

附錄C　ROM LISTING ·· 176

附錄D　GETTING APPLESOFT BASIC UP AND RUNNING ·· 219

附錄E　名詞解釋 ·· 223

附　　錄　　A

THE 6502 INSTRUCTIONS SET

6502 MICROPROCESSOR INSTRUCTIONS

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator.
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right one Bit (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

A	Accumulator
X, Y	Index Registers
M	Memory
$\bar{C}$	Borrow
P	Processor Status Register
S	Stack Pointer
✓	Change
—	No Change
+	Add
∧	Logical AND
-	Subtract
⊻	Logical Exclusive Or
↑	Transfer From Stack
↓	Transfer To Stack
→	Transfer To
←	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	Operand
#	Immediate Addressing Mode

FIGURE 1. ASL-SHIFT LEFT ONE BIT OPERATION

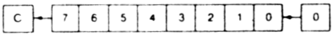


FIGURE 2. ROTATE ONE BIT LEFT (MEMORY OR ACCUMULATOR)

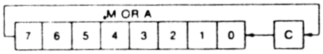
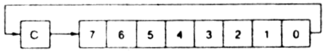


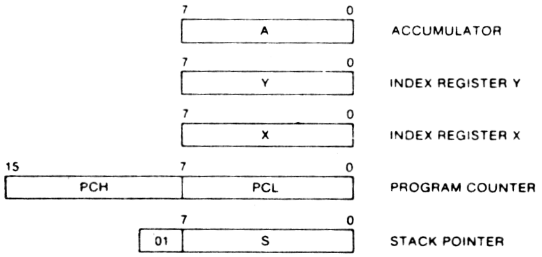
FIGURE 3

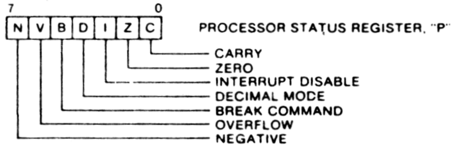


NOTE 1. BIT — TEST BITS

Bit 6 and 7 are transferred to the status register. If the result of A ∧ M is zero then Z=1, otherwise Z=0

PROGRAMMING MODEL





INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I D V
ADC						
Add memory to accumulator with carry	A+M+C → A,C	Immediate	ADC #Oper	69	2	√√√−−√
		Zero Page	ADC Oper	65	2	
		Zero Page,X	ADC Oper,X	75	2	
		Absolute	ADC Oper	6D	3	
		Absolute,X	ADC Oper,X	7D	3	
		Absolute,Y	ADC Oper,Y	79	3	
		(Indirect,X)	ADC (Oper,X)	61	2	
		(Indirect),Y	ADC (Oper),Y	71	2	
AND						
"AND" memory with accumulator	A∧M → A	Immediate	AND #Oper	29	2	√√−−−−
		Zero Page	AND Oper	25	2	
		Zero Page,X	AND Oper,X	35	2	
		Absolute	AND Oper	2D	3	
		Absolute,X	AND Oper,X	3D	3	
		Absolute,Y	AND Oper,Y	39	3	
		(Indirect,X)	AND (Oper,X)	21	2	
		(Indirect),Y	AND (Oper),Y	31	2	
ASL						
Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator	ASL A	0A	1	√√√−−−
		Zero Page	ASL Oper	06	2	
		Zero Page,X	ASL Oper,X	16	2	
		Absolute	ASL Oper	0E	3	
		Absolute,X	ASL Oper,X	1E	3	
BCC						
Branch on carry clear	Branch on C=0	Relative	BCC Oper	90	2	−−−−−−
BCS						
Branch on carry set	Branch on C=1	Relative	BCS Oper	B0	2	−−−−−−
BEQ						
Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	−−−−−−
BIT						
Test bits in memory with accumulator	A∧M, M_7 → N, M_6 → V	Zero Page	BIT* Oper	24	2	M_7√−−−M_6
		Absolute	BIT* Oper	2C	3	
BMI						
Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	−−−−−−
BNE						
Branch on result not zero	Branch on Z=0	Relative	BNE Oper	D0	2	−−−−−−
BPL						
Branch on result plus	Branch on N=0	Relative	BPL oper	10	2	−−−−−−
BRK						
Force Break	Forced Interrupt PC+2 ↓ P ↓	Implied	BRK*	00	1	−−−1−−
BVC						
Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	−−−−−−

Note 1: [illegible] 6 and 7 are transferred to the status register. If the result of A ∧ M is [illegible] then [illegible] 1, otherwise Z = 0

Note 2: A BRK command cannot be masked by setting I

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
BVS Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	------
CLC Clear carry flag	0 → C	Implied	CLC	18	1	---0--
CLD Clear decimal mode	0 → D	Implied	CLD	D8	1	-0----
CLI	0 → I	Implied	CLI	58	1	---0--
CLV Clear overflow flag	0 → V	Implied	CLV	B8	1	0-----
CMP Compare memory and accumulator	A − M	Immediate Zero Page Zero Page, X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y	CMP #Oper CMP Oper CMP Oper,X CMP Oper CMP Oper,X CMP Oper,Y CMP (Oper,X) CMP (Oper),Y	C9 C5 D5 CD DD D9 C1 D1	2 2 2 3 3 3 2 2	√√√---
CPX Compare memory and index X	X − M	Immediate Zero Page Absolute	CPX #Oper CPX Oper CPX Oper	E0 E4 EC	2 2 3	√√√---
CPY Compare memory and index Y	Y − M	Immediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper	C0 C4 CC	2 2 3	√√√---
DEC Decrement memory by one	M − 1 → M	Zero Page Zero Page,X Absolute Absolute,X	DEC Oper DEC Oper,X DEC Oper DEC Oper,X	C6 D6 CE DE	2 2 3 3	√√----
DEX Decrement index X by one	X − 1 → X	Implied	DEX	CA	1	√√----
DEY Decrement index Y by one	Y − 1 → Y	Implied	DEY	88	1	√√----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I D V
EOR						
"Exclusive-Or" memory with accumulator	A V M → A	Immediate	EOR #Oper	49	2	√√ - - - -
		Zero Page	EOR Oper	45	2	
		Zero Page,X	EOR Oper,X	55	2	
		Absolute	EOR Oper	4D	3	
		Absolute,X	EOR Oper,X	5D	3	
		Absolute,Y	EOR Oper,Y	59	3	
		(Indirect,X)	EOR (Oper,X)	41	2	
		(Indirect),Y	EOR (Oper),Y	51	2	
INC						
Increment memory by one	M + 1 → M	Zero Page	INC Oper	E6	2	√√ - - - -
		Zero Page,X	INC Oper,X	F6	2	
		Absolute	INC Oper	EE	3	
		Absolute,X	INC Oper,X	FE	3	
INX						
Increment index X by one	X + 1 → X	Implied	INX	E8	1	√√ - - - -
INY						
Increment index Y by one	Y + 1 → Y	Implied	INY	C8	1	√√ - - - -
JMP						
Jump to new location	(PC+1) → PCL	Absolute	JMP Oper	4C	3	- - - - - -
	(PC+2) → PCH	Indirect	JMP (Oper)	6C	3	
JSR						
Jump to new location saving return address	PC+2 ↓	Absolute	JSR Oper	20	3	- - - - - -
	(PC+1) → PCL					
	(PC+2) → PCH					
LDA						
Load accumulator with memory	M → A	Immediate	LDA #Oper	A9	2	√√ - - - -
		Zero Page	LDA Oper	A5	2	
		Zero Page,X	LDA Oper,X	B5	2	
		Absolute	LDA Oper	AD	3	
		Absolute,X	LDA Oper,X	BD	3	
		Absolute,Y	LDA Oper,Y	B9	3	
		(Indirect,X)	LDA (Oper,X)	A1	2	
		(Indirect),Y	LDA (Oper),Y	B1	2	
LDX						
Load index X with memory	M → X	Immediate	LDX #Oper	A2	2	√√ - - - -
		Zero Page	LDX Oper	A6	2	
		Zero Page,Y	LDX Oper,Y	B6	2	
		Absolute	LDX Oper	AE	3	
		Absolute,Y	LDX Oper,Y	BE	3	
LDY						
Load index Y with memory	M → Y	Immediate	LDY #Oper	A0	2	√√ - - - -
		Zero Page	LDY Oper	A4	2	
		Zero Page,X	LDY Oper,X	B4	2	
		Absolute	LDY Oper	AC	3	
		Absolute,X	LDY Oper,X	BC	3	

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No Bytes	"P" Status Reg N Z C I D V
LSR						
Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator	LSR A	4A	1	0√√---
		Zero Page	LSR Oper	46	2	
		Zero Page,X	LSR Oper,X	56	2	
		Absolute	LSR Oper	4E	3	
		Absolute,X	LSR Oper,X	5E	3	
NOP						
No operation	No Operation	Implied	NOP	EA	1	------
ORA						
"OR" memory with accumulator	A V M → A	Immediate	ORA #Oper	09	2	√√----
		Zero Page	ORA Oper	05	2	
		Zero Page,X	ORA Oper,X	15	2	
		Absolute	ORA Oper	0D	3	
		Absolute,X	ORA Oper,X	1D	3	
		Absolute,Y	ORA Oper,Y	19	3	
		(Indirect,X)	ORA (Oper,X)	01	2	
		(Indirect),Y	ORA (Oper),Y	11	2	
PHA						
Push accumulator on stack	A ↓	Implied	PHA	48	1	------
PHP						
Push processor status on stack	P ↓	Implied	PHP	08	1	------
PLA						
Pull accumulator from stack	A ↑	Implied	PLA	68	1	√√----
PLP						
Pull processor status from stack	P ↑	Implied	PLP	28	1	From Stack
ROL						
Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator	ROL A	2A	1	√√√---
		Zero Page	ROL Oper	26	2	
		Zero Page,X	ROL Oper,X	36	2	
		Absolute	ROL Oper	2E	3	
		Absolute,X	ROL Oper,X	3E	3	
ROR						
Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator	ROR A	6A	1	√√√---
		Zero Page	ROR Oper	66	2	
		Zero Page,X	ROR Oper,X	76	2	
		Absolute	ROR Oper	6E	3	
		Absolute,X	ROR Oper,X	7E	3	

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I D V
RTI Return from interrupt	P↑ PC↑	Implied	RTI	40	1	From Stack
RTS Return from subroutine	PC↑, PC+1 → PC	Implied	RTS	60	1	------
SBC Subtract memory from accumulator with borrow	A - M - $\overline{C}$ → A	Immediate	SBC #Oper	E9	2	√√√--√
		Zero Page	SBC Oper	E5	2	
		Zero Page,X	SBC Oper,X	F5	2	
		Absolute	SBC Oper	ED	3	
		Absolute,X	SBC Oper,X	FD	3	
		Absolute,Y	SBC Oper,Y	F9	3	
		(Indirect,X)	SBC (Oper,X)	E1	2	
		(Indirect),Y	SBC (Oper),Y	F1	2	
SEC Set carry flag	1 → C	Implied	SEC	38	1	--1---
SED Set decimal mode	1 → D	Implied	SED	F8	1	----1-
SEI Set interrupt disable status	1 → I	Implied	SEI	78	1	---1--
STA Store accumulator in memory	A → M	Zero Page	STA Oper	85	2	------
		Zero Page,X	STA Oper,X	95	2	
		Absolute	STA Oper	8D	3	
		Absolute,X	STA Oper,X	9D	3	
		Absolute,Y	STA Oper,Y	99	3	
		(Indirect,X)	STA (Oper,X)	81	2	
		(Indirect),Y	STA (Oper),Y	91	2	
STX Store index X in memory	X → M	Zero Page	STX Oper	86	2	------
		Zero Page,Y	STX Oper,Y	96	2	
		Absolute	STX Oper	8E	3	
STY Store index Y in memory	Y → M	Zero Page	STY Oper	84	2	------
		Zero Page,X	STY Oper,X	94	2	
		Absolute	STY Oper	8C	3	
TAX Transfer accumulator to index X	A → X	Implied	TAX	AA	1	√√----
TAY Transfer accumulator to index Y	A → Y	Implied	TAY	A8	1	√√----
TSX Transfer stack pointer to index X	S → X	Implied	TSX	BA	1	√√----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
TXA Transfer index X to accumulator	X → A	Implied	TXA	8A	1	√√----
TXS Transfer index X to stack pointer	X → S	Implied	TXS	9A	1	------
TYA Transfer index Y to accumulator	Y → A	Implied	TYA	98	1	√√----

HEX OPERATION CODES

00 — BRK
01 — ORA — (Indirect, X)
02 — NOP
03 — NOP
04 — NOP
05 — ORA — Zero Page
06 — ASL — Zero Page
07 — NOP
08 — PHP
09 — ORA — Immediate
0A — ASL — Accumulator
0B — NOP
0C — NOP
0D — ORA — Absolute
0E — ASL — Absolute
0F — NOP
10 — BPL
11 — ORA — (Indirect), Y
12 — NOP
13 — NOP
14 — NOP
15 — ORA — Zero Page, X
16 — ASL — Zero Page, X
17 — NOP
18 — CLC
19 — ORA — Absolute, Y
1A — NOP
1B — NOP
1C — NOP
1D — ORA — Absolute, X
1E — ASL — Absolute, X
1F — NOP
20 — JSR
21 — AND — (Indirect, X)
22 — NOP
23 — NOP
24 — BIT — Zero Page
25 — AND — Zero Page
26 — ROL — Zero Page
27 — NOP
28 — PLP
29 — AND — Immediate
2A — ROL — Accumulator
2B — NOP
2C — BIT — Absolute
2D — AND — Absolute
2E — ROL — Absolute
2F — NOP
30 — BMI
31 — AND — (Indirect), Y
32 — NOP
33 — NOP
34 — NOP
35 — AND — Zero Page, X
36 — ROL — Zero Page, X
37 — NOP
38 — SEC
39 — AND — Absolute, Y
3A — NOP
3B — NOP
3C — NOP
3D — AND — Absolute, X
3E — ROL — Absolute, X
3F — NOP
40 — RTI
41 — EOR — (Indirect, X)
42 — NOP
43 — NOP
44 — NOP
45 — EOR — Zero Page
46 — LSR — Zero Page
47 — NOP
48 — PHA
49 — EOR — Immediate
4A — LSR — Accumulator
4B — NOP
4C — JMP — Absolute
4D — EOR — Absolute
4E — LSR — Absolute
4F — NOP
50 — BVC
51 — EOR (Indirect), Y
52 — NOP
53 — NOP
54 — NOP
55 — EOR — Zero Page, X
56 — LSR — Zero Page, X
57 — NOP
58 — CLI
59 — EOR — Absolute, Y
5A — NOP
5B — NOP
5C — NOP
5D — EOR — Absolute, X
5E — LSR — Absolute, X
5F — NOP
60 — RTS
61 — ADC — (Indirect, X)
62 — NOP
63 — NOP
64 — NOP
65 — ADC — Zero Page
66 — ROR — Zero Page
67 — NOP
68 — PLA
69 — ADC — Immediate
6A — ROR — Accumulator
6B — NOP
6C — JMP — Indirect
6D — ADC — Absolute
6E — ROR — Absolute
6F — NOP
70 — BVS
71 — ADC — (Indirect), Y
72 — NOP
73 — NOP
74 — NOP
75 — ADC — Zero Page, X
76 — ROR — Zero Page, X
77 — NOP
78 — SEI
79 — ADC — Absolute, Y
7A — NOP
7B — NOP
7C — NOP
7D — ADC — Absolute, X NOP
7E — ROR — Absolute, X NOP
7F — NOP
80 — NOP
81 — STA — (Indirect, X)
82 — NOP
83 — NOP
84 —STY — Zero Page
85 — STA — Zero Page
86 — STX — Zero Page
87 — NOP
88 — DEY
89 — NOP
8A — TXA
8B — NOP
8C — STY — Absolute

8D — STA — Absolute
8E — STX — Absolute
8F — NOP
90 — BCC
91 — STA — (Indirect), Y
92 — NOP
93 — NOP
94 — STY — Zero Page, X
95 — STA — Zero Page, X
96 — STX — Zero Page, Y
97 — NOP
98 — TYA
99 — STA — Absolute, Y
9A — TXS
9B — NOP
9C — NOP
9D — STA — Absolute, X
9E — NOP
9F — NOP
A0 — LDY — Immediate
A1 — LDA — (Indirect, X)
A2 — LDX — Immediate
A3 — NOP
A4 — LDY — Zero Page
A5 — LDA — Zero Page
A6 — LDX — Zero Page
A7 — NOP
A8 — TAY
A9 — LDA — Immediate
AA — TAX
AB — NOP
AC — LDY — Absolute
AD — Absolute
AE — LDX — Absolute
AF — NOP
B0 — BCS
B1 — LDA — (Indirect), Y
B2 — NOP
B3 — NOP
B4 — LDY — Zero Page, X
B5 — LDA — Zero Page, X
B6 — LDX — Zero Page, Y
B7 — NOP
B8 — CLV
B9 — LDA — Absolute, Y
BA — TSX
BB — NOP
BC — LDY — Absolute, X
BD — LDA — Absolute, X
BE — LDX — Absolute, Y
BF — NOP
C0 — CPY — Immediate
C1 — CMP — (Indirect, X)
C2 — NOP
C3 — NOP
C4 — CPY — Zero Page
C5 — CMP — Zero Page
C6 — DEC — Zero Page
C7 — NOP
C8 — INY
C9 — CMP — Immediate
CA — DEX
CB — NOP
CC — CPY — Absolute
CD — CMP — Absolute
CE — DEC — Absolute
CF — NOP
D0 — BNE
D1 — CMP — (Indirect), Y
D2 — NOP
D3 — NOP
D4 — NOP
D5 — CMP — Zero Page, X
D6 — DEC — Zero Page, X
D7 — NOP
D8 — CLD
D9 — CMP — Absolute, Y
DA — NOP
DB — NOP
DC — NOP
DD — CMP — Absolute, X
DE — DEC — Absolute, X
DF — NOP
E0 — CPX — Immediate
E1 — SBC — (Indirect, X)
E2 — NOP
E3 — NOP
E4 — CPX — Zero Page
E5 — SBC — Zero Page
E6 — INC — Zero Page
E7 — NOP
E8 — INX
E9 — SBC — Immediate
EA — NOP
EB — NOP
EC — CPX — Absolute
ED — SBC — Absolute
EE — INC — Absolute
EF — NOP
F0 — BEQ
F1 — SBC — (Indirect), Y
F2 — NOP
F3 — NOP
F4 — NOP
F5 — SBC — Zero Page, X
F6 — INC — Zero Page, X
F7 — NOP
F8 — SED
F9 — SBC — Absolute, Y
FA — NOP
FB — NOP
FC — NOP
FD — SBC — Absolute, X
FE — INC — Absolute, X
FF — NOP

附　　錄　　B

SPECIAL LOCATIONS

Table 1: Keyboard Special Locations

Location: Hex	Decimal		Description:
$C000	49152	-16384	Keyboard Data
$C010	49168	-16368	Clear Keyboard Strobe

Table 4: Video Display Memory Ranges

Screen	Page	Begins at: Hex	Decimal	Ends at: Hex	Decimal
Text/Lo-Res	Primary	$400	1024	$7FF	2047
	Secondary	$800	2048	$BFF	3071
Hi-Res	Primary	$2000	8192	$3FFF	16383
	Secondary	$4000	16384	$5FFF	24575

Table 5: Screen Soft Switches

Location: Hex	Decimal		Description:
$C050	49232	-16304	Display a GRAPHICS mode.
$C051	49233	-16303	Display TEXT mode.
$C052	49234	-16302	Display all TEXT or GRAPHICS.
$C053	49235	-16301	Mix TEXT and a GRAPHICS mode.
$C054	49236	-16300	Display the Primary page (Page 1).
$C055	49237	-16299	Display the Secondary page (Page 2).
$C056	49238	-16298	Display LO-RES GRAPHICS mode.
$C057	49239	-16297	Display HI-RES GRAPHICS mode.

Table 9: Annunciator Special Locations

Ann.	State	Address: Decimal		Hex
0	off	49240	-16296	$C058
	on	49241	-16295	$C059
1	off	49242	-16294	$C05A
	on	49243	-16293	$C05B
2	off	49244	-16292	$C05C
	on	49245	-16291	$C05D
3	off	49246	-16290	$C05E
	on	49247	-16289	$C05F

Table 10: Input/Output Special Locations

Function	Address: Decimal		Hex	Read/Write
Speaker	49200	-16336	$C030	R
Cassette Out	49184	-16352	$C020	R
Cassette In	49256	-16288	$C060	R
Annunciators	49240 through 49247	-16296 through -16289	$C058 through $C05F	R/W
Flag inputs	49249	-16287	$C061	R
	49250	-16286	$C062	R
	49251	-16285	$C063	R
Analog Inputs	49252	-16284	$C064	R
	49253	-16283	$C065	
	49254	-16282	$C066	
	49255	-16281	$C067	
Analog Clear	49264	-16272	$C070	R/W
Utility Strobe	49216	-16320	$C040	R

Table 11: Text Window Special Locations

Function	Location: Decimal	Hex	Minimum/Normal/Maximum Value Decimal	Hex
Left Edge	32	$20	0/0/39	$0/$0/$17
Width	33	$21	0/40/40	$0/$28/$28
Top Edge	34	$22	0/0/24	$0/$0/$18
Bottom Edge	35	$23	0/24/24	$0/$18/$18

Table 12: Normal/Inverse Control Values

Value: Decimal	Hex	Effect:
255	$FF	COUT will display characters in Normal mode.
63	$3F	COUT will display characters in Inverse mode.
127	$7F	COUT will display letters in Flashing mode, all other characters in Inverse mode.

Table 13: Autostart ROM Special Locations

Location: Decimal	Hex	Contents:
1010 1011	$3F2 $3F3	Soft Entry Vector. These two locations contain the address of the reentry point for whatever language is in use. Normally contains $E003.
1012	$3F4	Power-Up Byte. Normally contains $45.
64367 (-1169)	$FB6F	This is the beginning of a machine language subroutine which sets up the power-up location.

Table 14: Page Three Monitor Locations			
Address: Decimal	**Hex**	**Use: Monitor ROM**	**Autostart ROM**
1008 1009	$3F0 $3F1	None.	Holds the address of the subroutine which handles machine language "BRK" requests (normaly $FA59).
1010 1011	$3F2 $3F3	None.	Soft Entry Vector.
1012	$3F4	None.	Power-up byte.
1013 1014 1015	$3F5 $3F6 $3F7	Holds a "JuMP" instruction to the subroutine which handles Applesoft II "&" commands. Normaly $4C $58 $FF.	
1016 1017 1018	$3F8 $3F9 $3FA	Holds a "JuMP" instruction to the subroutine which handles "User" (CTRL Y) commands.	
1019 1020 1021	$3FB $3FC $3FD	Holds a "JuMP" instruction to the subroutine which handles Non-Maskable Interrupts.	
1022 1023	$3FE $3FF	Holds the address of the subroutine which handles Interrupt ReQuests.	

Table 22: Built-In I/O Locations

	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$C000	Keyboard Data Input															
$C010	Clear Keyboard Strobe															
$C020	Cassette Output Toggle															
$C030	Speaker Toggle															
$C040	Utility Strobe															
$C050	gr	tx	nomix	mix	pri	sec	lores	hires	an0		an1		an2		an3	
$C060	cin	pb1	pb2	pb3	gc0	gc1	gc2	gc3	repeat $C060-$C067							
$C070	Game Controller Strobe															

Key to abbreviations:

gr	Set GRAPHICS mode	tx	Set TEXT mode
nomix	Set all text or graphics	mix	Mix text and graphics
pri	Display primary page	sec	Display secondary page
lores	Display Low-Res Graphics	hires	Display Hi-Res Graphics
an	Annunciator outputs	pb	Pushbutton inputs
gc	Game Controller inputs	cin	Cassette Input

Table 23: Peripheral Card I/O Locations																
	$Ø	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$CØ8Ø										Ø						
$CØ9Ø										1						
$CØAØ										2						
$CØBØ					Input/Output for slot number					3						
$CØCØ										4						
$CØDØ										5						
$CØEØ										6						
$CØFØ										7						

Table 24: Peripheral Card PROM Locations																
	$ØØ	$1Ø	$2Ø	$3Ø	$4Ø	$5Ø	$6Ø	$7Ø	$8Ø	$9Ø	$AØ	$BØ	$CØ	$DØ	$EØ	$FØ
$C1ØØ										1						
$C2ØØ										2						
$C3ØØ										3						
$C4ØØ					PROM space for slot number					4						
$C5ØØ										5						
$C6ØØ										6						
$C7ØØ										7						

Table 25: I/O Location Base Addresses								
Base	Slot							
Address	Ø	1	2	3	4	5	6	7
$CØ8Ø	$CØ8Ø	$CØ9Ø	$CØAØ	$CØBØ	$CØCØ	$CØDØ	$CØEØ	$CØFØ
$CØ81	$CØ81	$CØ91	$CØA1	$CØB1	$CØC1	$CØD1	$CØE1	$CØF1
$CØ82	$CØ82	$CØ92	$CØA2	$CØB2	$CØC2	$CØD2	$CØE2	$CØF2
$CØ83	$CØ83	$CØ93	$CØA3	$CØB3	$CØC3	$CØD3	$CØE3	$CØF3
$CØ84	$CØ84	$CØ94	$CØA4	$CØB4	$CØC4	$CØD4	$CØE4	$CØF4
$CØ85	$CØ85	$CØ95	$CØA5	$CØB5	$CØC5	$CØD5	$CØE5	$CØF5
$CØ86	$CØ86	$CØ96	$CØA6	$CØB6	$CØC6	$CØD6	$CØE6	$CØF6
$CØ87	$CØ87	$CØ97	$CØA7	$CØB7	$CØC7	$CØD7	$CØE7	$CØF7
$CØ88	$CØ88	$CØ98	$CØA8	$CØB8	$CØC8	$CØD8	$CØE8	$CØF8
$CØ89	$CØ89	$CØ99	$CØA9	$CØB9	$CØC9	$CØD9	$CØE9	$CØF9
$CØ8A	$CØ8A	$CØ9A	$CØAA	$CØBA	$CØCA	$CØDA	$CØEA	$CØFA
$CØ8B	$CØ8B	$CØ9B	$CØAB	$CØBB	$CØCB	$CØDB	$CØEB	$CØFB
$CØ8C	$CØ8C	$CØ9C	$CØAC	$CØBC	$CØCC	$CØDC	$CØEC	$CØFC
$CØ8D	$CØ8D	$CØ9D	$CØAD	$CØBD	$CØCD	$CØDD	$CØED	$CØFD
$CØ8E	$CØ8E	$CØ9E	$CØAE	$CØBE	$CØCE	$CØDE	$CØEE	$CØFE
$CØ8F	$CØ8F	$CØ9F	$CØAF	$CØBF	$CØCF	$CØDF	$CØEF	$CØFF
	I/O Locations							

Table 26: I/O Scratchpad RAM Addresses

Base Address	Slot Number 1	2	3	4	5	6	7
$0478	$0479	$047A	$047B	$047C	$047D	$047E	$047F
$04F8	$04F9	$04FA	$04FB	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B	$057C	$057D	$057E	$057F
$05F8	$05F9	$05FA	$05FB	$05FC	$05FD	$05FE	$05FF
$0678	$0679	$067A	$067B	$067C	$067D	$067E	$067F
$06F8	$06F9	$06FA	$06FB	$06FC	$06FD	$06FE	$06FF
$0778	$0779	$077A	$077B	$077C	$077D	$077E	$077F
$07F8	$07F9	$07FA	$07FB	$07FC	$07FD	$07FE	$07FF

附　錄　C

ROM LISTING

AUTOSTART ROM LISTING ······························ 177

MONITOR ROM LISTING ······························ 196

AUTOSTART ROM LISTING

```
0000:        ********************************
0000:        *
0000:        * APPLE II
0000:        * MONITOR II
0000:        *
0000:        * COPYRIGHT 1978 BY
0000:        * APPLE COMPUTER, INC.
0000:        *
0000:        * ALL RIGHTS RESERVED
0000:        *
0000:        * STEVE WOZNIAK
0000:        *
0000:        ********************************
0000:        *
0000:        * MODIFIED NOV 1978
0000:        * BY JOHN A
0000:        *
0000:        ********************************
F800:                ORG $F800
F800:                OBJ $2000
F800:        ******************************
F800:        LOC0     EQU $00
F800:        LOC1     EQU $01
F800:        WNDLFT   EQU $20
F800:        WNDWDTH  EQU $21
F800:        WNDTOP   EQU $22
F800:        WNDBTM   EQU $23
F800:        CH       EQU $24
F800:        CV       EQU $25
F800:        GBASL    EQU $26
F800:        GBASH    EQU $27
F800:        BASL     EQU $28
F800:        BASH     EQU $29
F800:        BAS2L    EQU $2A
F800:        BAS2H    EQU $2B
F800:        H2       EQU $2C
F800:        LMNEM    EQU $2C
F800:        V2       EQU $2D
F800:        RMNEM    EQU $2D
F800:        MASK     EQU $2E
F800:        CHKSUM   EQU $2E
F800:        FORMAT   EQU $2E
F800:        LASTIN   EQU $2F
F800:        LENGTH   EQU $2F
F800:        SIGN     EQU $2F
F800:        COLOR    EQU $30
F800:        MODE     EQU $31
F800:        INVFLG   EQU $32
F800:        PROMPT   EQU $33
F800:        YSAV     EQU $34
F800:        YSAV1    EQU $35
F800:        CSWL     EQU $36
F800:        CSWH     EQU $37
F800:        KSWL     EQU $38
F800:        KSWH     EQU $39
F800:        PCL      EQU $3A
F800:        PCH      EQU $3B
F800:        A1L      EQU $3C
F800:        A1H      EQU $3D
F800:        A2L      EQU $3E
F800:        A2H      EQU $3F
F800:        A3L      EQU $40
F800:        A3H      EQU $41
F800:        A4L      EQU $42
F800:        A4H      EQU $43
F800:        A5L      EQU $44
F800:        A5H      EQU $45
```

```
F800:                ACC      EQU $45      ; NOTE OVERLAP WITH A5H!
F800:                XREG     EQU $46
F800:                YREG     EQU $47
F800:                STATUS   EQU $48
F800:                SPNT     EQU $49
F800:                RNDL     EQU $4E
F800:                RNDH     EQU $4F
F800:                PICK     EQU $95
F800:                IN       EQU $0200
F800:                BRKV     EQU $3F0     ; NEW VECTOR FOR BRK
F800:                SOFTEV   EQU $3F2     ; VECTOR FOR WARM START
F800:                PWREDUP  EQU $3F4     ; THIS MUST = EOR #$A5 OF SOFTEV+1
F800:                AMPERV   EQU $3F5     ; APPLESOFT & EXIT VECTOR
F800:                USRADR   EQU $03F8
F800:                NMI      EQU $03FB
F800:                IRQLOC   EQU $3FE
F800:                LINE1    EQU $400
F800:                MSLOT    EQU $07F8
F800:                IOADR    EQU $C000
F800:                KBD      EQU $C000
F800:                KBDSTRB  EQU $C010
F800:                TAPEOUT  EQU $C020
F800:                SPKR     EQU $C030
F800:                TXTCLR   EQU $C050
F800:                TXTSET   EQU $C051
F800:                MIXCLR   EQU $C052
F800:                MIXSET   EQU $C053
F800:                LOWSCR   EQU $C054
F800:                HISCR    EQU $C055
F800:                LORES    EQU $C056
F800:                HIRES    EQU $C057
F800:                SETAN0   EQU $C058
F800:                CLRAN0   EQU $C059
F800:                SETAN1   EQU $C05A
F800:                CLRAN1   EQU $C05B
F800:                SETAN2   EQU $C05C
F800:                CLRAN2   EQU $C05D
F800:                SETAN3   EQU $C05E
F800:                CLRAN3   EQU $C05F
F800:                TAPEIN   EQU $C060
F800:                PADDL0   EQU $C064
F800:                PTRIG    EQU $C070
F800:                CLRROM   EQU $CFFF
F800:                BASIC    EQU $E000
F800:                BASIC2   EQU $E003
F800:                         PAGE
F800:  4A            PLOT     LSR A
F801:  08                     PHP
F802:  20 47 F8               JSR GBASCALC
F805:  28                     PLP
F806:  A9 0F                  LDA #$0F
F808:  90 02                  BCC RTMASK
F80A:  69 E0                  ADC #$E0
F80C:  85 2E         RTMASK   STA MASK
F80E:  B1 26         PLOT1    LDA (GBASL),Y
F810:  45 30                  EOR COLOR
F812:  25 2E                  AND MASK
F814:  51 26                  EOR (GBASL),Y
F816:  91 26                  STA (GBASL),Y
F818:  60                     RTS
F819:  20 00 F8      HLINE    JSR PLOT
F81C:  C4 2C         HLINE1   CPY H2
F81E:  B0 11                  BCS RTS1
F820:  C8                     INY
F821:  20 0E F8               JSR PLOT1
F824:  90 F6                  BCC HLINE1
F826:  69 01         VLINEZ   ADC #$01
F828:  48            VLINE    PHA
F829:  20 00 F8               JSR PLOT
F82C:  68                     PLA
F82D:  C5 2D                  CMP V2
F82F:  90 F5                  BCC VLINEZ
F831:  60            RTS1     RTS
```

```
F832:  A0 2F      CLRSCR   LDY #$2F
F834:  D0 02               BNE CLRSC2
F836:  A0 27      CLRTOP   LDY #$27
F838:  84 2D      CLRSC2   STY V2
F83A:  A0 27               LDY #$27
F83C:  A9 00      CLRSC3   LDA #$00
F83E:  85 30               STA COLOR
F840:  20 28 F8            JSR VLINE
F843:  88                  DEY
F844:  10 F6               BPL CLRSC3
F846:  60                  RTS
F847:                      PAGE
F847:  48         GBASCALC PHA
F848:  4A                  LSR A
F849:  29 03               AND #$03
F84B:  09 04               ORA #$04
F84D:  85 27               STA GBASH
F84F:  68                  PLA
F850:  29 18               AND #$18
F852:  90 02               BCC GBCALC
F854:  69 7F               ADC #$7F
F856:  85 26      GBCALC   STA GBASL
F858:  0A                  ASL A
F859:  0A                  ASL A
F85A:  05 26               ORA GBASL
F85C:  85 26               STA GBASL
F85E:  60                  RTS
F85F:  A5 30               LDA COLOR
F861:  18                  CLC
F862:  69 03               ADC #$03
F864:  29 0F      SETCOL   AND #$0F
F866:  85 30               STA COLOR
F868:  0A                  ASL A
F869:  0A                  ASL A
F86A:  0A                  ASL A
F86B:  0A                  ASL A
F86C:  05 30               ORA COLOR
F86E:  85 30               STA COLOR
F870:  60                  RTS
F871:  4A         SCRN     LSR A
F872:  08                  PHP
F873:  20 47 F8            JSR GBASCALC
F876:  B1 26               LDA (GBASL),Y
F878:  28                  PLP
F879:  90 04      SCRN2    BCC RTMSKZ
F87B:  4A                  LSR A
F87C:  4A                  LSR A
F87D:  4A                  LSR A
F87E:  4A                  LSR A
F87F:  29 0F      RTMSKZ   AND #$0F
F881:  60                  RTS
F882:                      PAGE
F882:  A6 3A      INSDS1   LDX PCL
F884:  A4 3B               LDY PCH
F886:  20 96 FD            JSR PRYX2
F889:  20 48 F9            JSR PRBLNK
F88C:  A1 3A      INSDS2   LDA (PCL,X)
F88E:  A8                  TAY
F88F:  4A                  LSR A
F890:  90 09               BCC IEVEN
F892:  6A                  ROR A
F893:  B0 10               BCS ERR
F895:  C9 A2               CMP #$A2
F897:  F0 0C               BEQ ERR
F899:  29 87               AND #$87
F89B:  4A         IEVEN    LSR A
F89C:  AA                  TAX
F89D:  BD 62 F9            LDA FMT1,X
F8A0:  20 79 F8            JSR SCRN2
F8A3:  D0 04               BNE GETFMT
F8A5:  A0 80      ERR      LDY #$80
F8A7:  A9 00               LDA #$00
F8A9:  AA         GETFMT   TAX
```

```
F8AA:  BD A6 F9             LDA FMT2,X
F8AD:  85 2E                STA FORMAT
F8AF:  29 03                AND #$03
F8B1:  85 2F                STA LENGTH
F8B3:  98                   TYA
F8B4:  29 8F                AND #$8F
F8B6:  AA                   TAX
F8B7:  98                   TYA
F8B8:  A0 03                LDY #$03
F8BA:  E0 8A                CPX #$8A
F8BC:  F0 0B                BEQ MNNDX3
F8BE:  4A         MNNDX1    LSR A
F8BF:  90 08                BCC MNNDX3
F8C1:  4A                   LSR A
F8C2:  4A         MNNDX2    LSR A
F8C3:  09 20                ORA #$20
F8C5:  88                   DEY
F8C6:  D0 FA                BNE MNNDX2
F8C8:  C8                   INY
F8C9:  88         MNNDX3    DEY
F8CA:  D0 F2                BNE MNNDX1
F8CC:  60                   RTS
F8CD:  FF FF FF             DFB $FF,$FF,$FF
F8D0                        PAGE
F8D0:  20 82 F8   INSTDSP   JSR INSDS1
F8D3:  48                   PHA
F8D4:  B1 3A      PRNTOP    LDA (PCL),Y
F8D6:  20 DA FD             JSR PRBYTE
F8D9:  A2 01                LDX #$01
F8DB:  20 4A F9   PRNTBL    JSR PRBL2
F8DE:  C4 2F                CPY LENGTH
F8E0:  C8                   INY
F8E1:  90 F1                BCC PRNTOP
F8E3:  A2 03                LDX #$03
F8E5:  C0 04                CPY #$04
F8E7:  90 F2                BCC PRNTBL
F8E9:  68                   PLA
F8EA:  A8                   TAY
F8EB:  B9 C0 F9             LDA MNEML,Y
F8EE:  85 2C                STA LMNEM
F8F0:  B9 00 FA             LDA MNEMR,Y
F8F3:  85 2D                STA RMNEM
F8F5:  A9 00      NXTCOL    LDA #$00
F8F7:  A0 05                LDY #$05
F8F9:  06 2D      PRMN2     ASL RMNEM
F8FB:  26 2C                ROL LMNEM
F8FD:  2A                   ROL A
F8FE:  88                   DEY
F8FF:  D0 F8                BNE PRMN2
F901:  69 BF                ADC #$BF
F903:  20 ED FD             JSR COUT
F906:  CA                   DEX
F907:  D0 EC                BNE NXTCOL
F909:  20 48 F9             JSR PRBLNK
F90C:  A4 2F                LDY LENGTH
F90E:  A2 06                LDX #$06
F910:  E0 03      PRADR1    CPX #$03
F912:  F0 1C                BEQ PRADR5
F914:  06 2E      PRADR2    ASL FORMAT
F916:  90 0E                BCC PRADR3
F918:  BD B3 F9             LDA CHAR1-1,X
F91B:  20 ED FD             JSR COUT
F91E:  BD B9 F9             LDA CHAR2-1,X
F921:  F0 03                BEQ PRADR3
F923:  20 ED FD             JSR COUT
F926:  CA         PRADR3    DEX
F927:  D0 E7                BNE PRADR1
F929:  60                   RTS
F92A:  88         PRADR4    DEY
F92B:  30 E7                BMI PRADR2
F92D:  20 DA FD             JSR PRBYTE
F930:  A5 2E      PRADR5    LDA FORMAT
F932:  C9 E8                CMP #$E8
```

```
F934:  B1 3A                LDA (PCL),Y
F936:  90 F2                BCC PRADR4
F938:                       PAGE
F938:  20 56 F9    RELADR   JSR PCADJ3
F93B:  AA                   TAX
F93C:  E8                   INX
F93D:  D0 01                BNE PRNTYX
F93F:  C8                   INY
F940:  98          PRNTYX   TYA
F941:  20 DA FD    PRNTAX   JSR PRBYTE
F944:  8A          PRNTX    TXA
F945:  4C DA FD             JMP PRBYTE
F948:  A2 03       PRBLNK   LDX #$03
F94A:  A9 A0       PRBL2    LDA #$A0
F94C:  20 ED FD    PRBL3    JSR COUT
F94F:  CA                   DEX
F950:  D0 F8                BNE PRBL2
F952:  60                   RTS
F953:  38          PCADJ    SEC
F954:  A5 2F       PCADJ2   LDA LENGTH
F956:  A4 3B       PCADJ3   LDY PCH
F958:  AA                   TAX
F959:  10 01                BPL PCADJ4
F95B:  88                   DEY
F95C:  65 3A       PCADJ4   ADC PCL
F95E:  90 01                BCC RTS2
F960:  C8                   INY
F961:  60          RTS2     RTS
F962:  04          FMT1     DFB $04
F963:  20                   DFB $20
F964:  54                   DFB $54
F965:  30                   DFB $30
F966:  0D                   DFB $0D
F967:  80                   DFB $80
F968:  04                   DFB $04
F969:  90                   DFB $90
F96A:  03                   DFB $03
F96B:  22                   DFB $22
F96C:  54                   DFB $54
F96D:  33                   DFB $33
F96E:  0D                   DFB $0D
F96F:  80                   DFB $80
F970:  04                   DFB $04
F971:  90                   DFB $90
F972:  04                   DFB $04
F973:  20                   DFB $20
F974:  54                   DFB $54
F975:  33                   DFB $33
F976:  0D                   DFB $0D
F977:  80                   DFB $80
F978:  04                   DFB $04
F979:  90                   DFB $90
F97A:  04                   DFB $04
F97B:  20                   DFB $20
F97C:  54                   DFB $54
F97D:  3B                   DFB $3B
F97E:  0D                   DFB $0D
F97F:  80                   DFB $80
F980:  04                   DFB $04
F981:  90                   DFB $90
F982:  00                   DFB $00
F983:  22                   DFB $22
F984:  44                   DFB $44
F985:  33                   DFB $33
F986:  0D                   DFB $0D
F987:  C8                   DFB $C8
F988:  44                   DFB $44
F989:  00                   DFB $00
F98A:  11                   DFB $11
F98B:  22                   DFB $22
F98C:  44                   DFB $44
F98D:  33                   DFB $33
F98E:  0D                   DFB $0D
```

```
F98F:  C8     361         DFB $C8
F990:  44     362         DFB $44
F991:  A9     363         DFB $A9
F992:  01     364         DFB $01
F993:  22     365         DFB $22
F994:  44     366         DFB $44
F995:  33     367         DFB $33
F996:  0D     368         DFB $0D
F997:  80     369         DFB $80
F998:  04     370         DFB $04
F999:  90     371         DFB $90
F99A:  01     372         DFB $01
F99B:  22     373         DFB $22
F99C:  44     374         DFB $44
F99D:  33     375         DFB $33
F99E:  0D     376         DFB $0D
F99F:  80     377         DFB $80
F9A0:  04     378         DFB $04
F9A1:  90     379         DFB $90
F9A2:  26     380         DFB $26
F9A3:  31     381         DFB $31
F9A4:  87     382         DFB $87
F9A5:  9A     383         DFB $9A
F9A6:  00     384 FMT2    DFB $00
F9A7:  21     385         DFB $21
F9A8:  81     386         DFB $81
F9A9:  82     387         DFB $82
F9AA:  00     388         DFB $00
F9AB:  00     389         DFB $00
F9AC:  59     390         DFB $59
F9AD:  4D     391         DFB $4D
F9AE:  91     392         DFB $91
F9AF:  92     393         DFB $92
F9B0:  86     394         DFB $86
F9B1:  4A     395         DFB $4A
F9B2:  85     396         DFB $85
F9B3:  9D     397         DFB $9D
F9B4:  AC     398 CHAR1   DFB $AC
F9B5:  A9     399         DFB $A9
F9B6:  AC     400         DFB $AC
F9B7:  A3     401         DFB $A3
F9B8:  A8     402         DFB $A8
F9B9:  A4     403         DFB $A4
F9BA:  D9     404 CHAR2   DFB $D9
F9BB:  00     405         DFB $00
F9BC:  D8     406         DFB $D8
F9BD:  A4     407         DFB $A4
F9BE:  A4     408         DFB $A4
F9BF:  00     409         DFB $00
F9C0:  1C     410 MNEML   DFB $1C
F9C1:  8A     411         DFB $8A
F9C2:  1C     412         DFB $1C
F9C3:  23     413         DFB $23
F9C4:  5D     414         DFB $5D
F9C5:  8B     415         DFB $8B
F9C6:  1B     416         DFB $1B
F9C7:  A1     417         DFB $A1
F9C8:  9D     418         DFB $9D
F9C9:  8A     419         DFB $8A
F9CA:  1D     420         DFB $1D
F9CB:  23     421         DFB $23
F9CC:  9D     422         DFB $9D
F9CD:  8B     423         DFB $8B
F9CE:  1D     424         DFB $1D
F9CF:  A1     425         DFB $A1
F9D0:  00     426         DFB $00
F9D1:  29     427         DFB $29
F9D2:  19     428         DFB $19
F9D3:  AE     429         DFB $AE
F9D4:  69     430         DFB $69
F9D5:  A8     431         DFB $A8
F9D6:  19     432         DFB $19
F9D7:  23     433         DFB $23
```

```
F9D8:  24                 DFB  $24
F9D9:  53                 DFB  $53
F9DA:  1B                 DFB  $1B
F9DB:  23                 DFB  $23
F9DC:  24                 DFB  $24
F9DD:  53                 DFB  $53
F9DE:  19                 DFB  $19
F9DF:  A1                 DFB  $A1
F9E0:  00                 DFB  $00
F9E1:  1A                 DFB  $1A
F9E2:  5B                 DFB  $5B
F9E3:  5B                 DFB  $5B
F9E4:  A5                 DFB  $A5
F9E5:  69                 DFB  $69
F9E6:  24                 DFB  $24
F9E7:  24                 DFB  $24
F9E8:  AE                 DFB  $AE
F9E9:  AE                 DFB  $AE
F9EA:  A8                 DFB  $A8
F9EB:  AD                 DFB  $AD
F9EC:  29                 DFB  $29
F9ED:  00                 DFB  $00
F9EE:  7C                 DFB  $7C
F9EF:  00                 DFB  $00
F9F0:  15                 DFB  $15
F9F1:  9C                 DFB  $9C
F9F2:  6D                 DFB  $6D
F9F3:  9C                 DFB  $9C
F9F4:  A5                 DFB  $A5
F9F5:  69                 DFB  $69
F9F6:  29                 DFB  $29
F9F7:  53                 DFB  $53
F9F8:  84                 DFB  $84
F9F9:  13                 DFB  $13
F9FA:  34                 DFB  $34
F9FB:  11                 DFB  $11
F9FC:  A5                 DFB  $A5
F9FD:  69                 DFB  $69
F9FE:  23                 DFB  $23
F9FF:  A0                 DFB  $A0
FA00:  D8         MNEMR   DFB  $D8
FA01:  62                 DFB  $62
FA02:  5A                 DFB  $5A
FA03:  48                 DFB  $48
FA04:  26                 DFB  $26
FA05:  62                 DFB  $62
FA06:  94                 DFB  $94
FA07:  88                 DFB  $88
FA08:  54                 DFB  $54
FA09:  44                 DFB  $44
FA0A:  C8                 DFB  $C8
FA0B:  54                 DFB  $54
FA0C:  68                 DFB  $68
FA0D:  44                 DFB  $44
FA0E:  E8                 DFB  $E8
FA0F:  94                 DFB  $94
FA10:  00                 DFB  $00
FA11:  B4                 DFB  $B4
FA12:  08                 DFB  $08
FA13:  84                 DFB  $84
FA14:  74                 DFB  $74
FA15:  B4                 DFB  $B4
FA16:  28                 DFB  $28
FA17:  6E                 DFB  $6E
FA18:  74                 DFB  $74
FA19:  F4                 DFB  $F4
FA1A:  CC                 DFB  $CC
FA1B:  4A                 DFB  $4A
FA1C:  72                 DFB  $72
FA1D:  F2                 DFB  $F2
FA1E:  A4                 DFB  $A4
FA1F:  8A                 DFB  $8A
FA20:  00                 DFB  $00
```

```
FA21:  AA                 DFB $AA
FA22:  A2                 DFB $A2
FA23:  A2                 DFB $A2
FA24:  74                 DFB $74
FA25:  74                 DFB $74
FA26:  74                 DFB $74
FA27:  72                 DFB $72
FA28:  44                 DFB $44
FA29:  68                 DFB $68
FA2A:  B2                 DFB $B2
FA2B:  32                 DFB $32
FA2C:  B2                 DFB $B2
FA2D:  00                 DFB $00
FA2E:  22                 DFB $22
FA2F:  00                 DFB $00
FA30:  1A                 DFB $1A
FA31:  1A                 DFB $1A
FA32:  26                 DFB $26
FA33:  26                 DFB $26
FA34:  72                 DFB $72
FA35:  72                 DFB $72
FA36:  88                 DFB $88
FA37:  C8                 DFB $C8
FA38:  C4                 DFB $C4
FA39:  CA                 DFB $CA
FA3A:  26                 DFB $26
FA3B:  48                 DFB $48
FA3C:  44                 DFB $44
FA3D:  44                 DFB $44
FA3E:  A2                 DFB $A2
FA3F:  C8                 DFB $C8
FA40:                     PAGE
FA40:  85 45      IRQ     STA ACC
FA42:  68                 PLA
FA43:  48                 PHA
FA44:  0A                 ASL A
FA45:  0A                 ASL A
FA46:  0A                 ASL A
FA47:  30 03              BMI BREAK
FA49:  6C FE 03           JMP (IRQLOC)
FA4C:  28         BREAK   PLP
FA4D:  20 4C FF           JSR SAV1
FA50:  68                 PLA
FA51:  85 3A              STA PCL
FA53:  68                 PLA
FA54:  85 3B              STA PCH
FA56:  6C F0 03           JMP (BRKV) ;BRKV WRITTEN OVER BY DISK BOOT
FA59:  20 82 F8   OLDBRK  JSR INSDS1
FA5C:  20 DA FA           JSR RGDSP1
FA5F:  4C 65 FF           JMP MON
FA62:  D8         RESET   CLD         ;  DO THIS FIRST THIS TIME
FA63:  20 84 FE           JSR SETNORM
FA66:  20 2F FB           JSR INIT
FA69:  20 93 FE           JSR SETVID
FA6C:  20 89 FE           JSR SETKBD
FA6F:  AD 58 C0   INITAN  LDA SETAN0 ;  AN0 = TTL HI
FA72:  AD 5A C0           LDA SETAN1 ;  AN1 = TTL HI
FA75:  AD 5D C0           LDA CLRAN2 ;  AN2 = TTL LO
FA78:  AD 5F C0           LDA CLRAN3 ;  AN3 = TTL LO
FA7B:  AD FF CF           LDA CLRROM ;  TURN OFF EXTNSN ROM
FA7E:  2C 10 C0           BIT KBDSTRB ;  CLEAR KEYBOARD
FA81:  D8         NEWMON  CLD
FA82:  20 3A FF           JSR BELL    ;  CAUSES DELAY IF KEY BOUNCES
FA85:  AD F3 03           LDA SOFTEV+1 ;IS RESET HI
FA88:  49 A5              EOR #$A5    ;A FUNNY COMPLEMENT OF THE
FA8A:  CD F4 03           CMP PWREDUP ;  PWR UP BYTE ???
FA8D:  D0 17              BNE PWRUP   ;  NO SO PWRUP
FA8F:  AD F2 03           LDA SOFTEV  ;  YES SEE IF COLD START
FA92:  D0 0F              BNE NOFIX   ;  HAS BEEN DONE YET?
FA94:  A9 E0              LDA #$E0    ;  ??
FA96:  CD F3 03           CMP SOFTEV+1 ;  ??
FA99:  D0 08              BNE NOFIX   ;  YES SO REENTER SYSTEM
FA9B:  A0 03      FIXSEV  LDY #3      ;  NO SO POINT AT WARM START
```

```
FA9D: 8C F2 03             STY SOFTEV  ; FOR NEXT RESET
FAA0: 4C 00 E0             JMP BASIC   ; AND DO THE COLD START
FAA3: 6C F2 03    NOFIX    JMP (SOFTEV) ; SOFT ENTRY VECTOR
FAA6:             *****************************
FAA6: 20 60 FB    PWRUP    JSR APPLEII
FAA9:             SETPG3   EQU *       ; SET PAGE 3 VECTORS
FAA9: A2 05                LDX #5
FAAB: BD FC FA    SETPLP   LDA PWRCON-1,X ; WITH CNTRL B ADRS
FAAE: 9D EF 03             STA BRKV-1,X ; OF CURRENT BASIC
FAB1: CA                   DEX
FAB2: D0 F7                BNE SETPLP
FAB4: A9 C8                LDA #$C8    ; LOAD HI SLOT +1
FAB6: 86 00                STX LOC0    ; SETPG3 MUST RETURN X=0
FAB8: 85 01                STA LOC1    ; SET PTR H
FABA: A0 07       SLOOP    LDY #7      ; Y IS BYTE PTR
FABC: C6 01                DEC LOC1
FABE: A5 01                LDA LOC1
FAC0: C9 C0                CMP #$C0    ; AT LAST SLOT YET?
FAC2: F0 D7                BEQ FIXSEV  ; YES AND IT CANT BE A DISK
FAC4: 8D F8 07             STA MSLOT
FAC7: B1 00       NXTBYT   LDA (LOC0),Y ; FETCH A SLOT BYTE
FAC9: D9 01 FB             CMP DISKID-1,Y ; IS IT A DISK ??
FACC: D0 EC                BNE SLOOP   ; NO SO NEXT SLOT DOWN
FACE: 88                   DEY
FACF: 88                   DEY         ; YES SO CHECK NEXT BYTE
FAD0: 10 F5                BPL NXTBYT  ; UNTIL 4 CHECKED
FAD2: 6C 00 00             JMP (LOC0)
FAD5: EA                   NOP
FAD6: EA                   NOP
FAD7:             * REGDSP MUST ORG $FAD7
FAD7: 20 8E FD    REGDSP   JSR CROUT
FADA: A9 45       RGDSP1   LDA #$45
FADC: 85 40                STA A3L
FADE: A9 00                LDA #$00
FAE0: 85 41                STA A3H
FAE2: A2 FB                LDX #$FB
FAE4: A9 A0       RDSP1    LDA #$A0
FAE6: 20 ED FD             JSR COUT
FAE9: BD 1E FA             LDA RTBL-251,X
FAEC: 20 ED FD             JSR COUT
FAEF: A9 BD                LDA #$BD
FAF1: 20 ED FD             JSR COUT
FAF4:             * LDA ACC+5,X
FAF4: B5 4A                DFB $B5,$4A
FAF6: 20 DA FD             JSR PRBYTE
FAF9: E8                   INX
FAFA: 30 E8                BMI RDSP1
FAFC: 60                   RTS
FAFD: 59 FA       PWRCON   DW  OLDBRK
FAFF: 00 E0 45             DFB $00,$E0,$45
FB02: 20 FF 00
FB05: FF          DISKID   DFB $20,$FF,$00,$FF
FB06: 03 FF 3C             DFB $03,$FF,$3C
FB09: C1 D0 D0    TITLE    DFB $C1,$D0,$D0
FB0C: CC C5 A0             DFB $CC,$C5,$A0
FB0F: DD DB                DFB $DD,$DB
FB11:             XLTBL    EQU *
FB11: C4 C2 C1             DFB $C4,$C2,$C1
FB14: FF C3                DFB $FF,$C3
FB16: FF FF FF             DFB $FF,$FF,$FF
FB19:             *  MUST ORG $FB19
FB19: C1 D8 D9    RTBL     DFB $C1,$D8,$D9
FB1C: D0 D3                DFB $D0,$D3
FB1E: AD 70 C0    PREAD    LDA PTRIG
FB21:                      LST ON
FB21: A0 00                LDY #$00
FB23: EA                   NOP
FB24: EA                   NOP
FB25: BD 64 C0    PREAD2   LDA PADDLO,X
FB28: 10 04                BPL RTS2D
FB2A: C8                   INY
FB2B: D0 F8                BNE PREAD2
FB2D: 88                   DEY
```

```
FB2E: 60              RTS2D    RTS
FB2F: A9 00           INIT     LDA #$00
FB31: 85 48                    STA STATUS
FB33: AD 56 C0                 LDA LORES
FB36: AD 54 C0                 LDA LOWSCR
FB39: AD 51 C0        SETTXT   LDA TXTSET
FB3C: A9 00                    LDA #$00
FB3E: F0 0D                    BEQ SETWND
FB40: AD 50 C0        SETGR    LDA TXTCLR
FB43: AD 53 C0                 LDA MIXSET
FB46: 20 36 F8                 JSR CLRTOP
FB49: A9 14                    LDA #$14
FB4B: 85 22           SETWND   STA WNDTOP
FB4D: A9 00                    LDA #$00
FB4F: 85 20                    STA WNDLFT
FB51: A9 28                    LDA #$28
FB53: 85 21                    STA WNDWDTH
FB55: A9 18                    LDA #$18
FB57: 85 23                    STA WNDBTM
FB59: A9 17                    LDA #$17
FB5B: 85 25           TABV     STA CV
FB5D: 4C 22 FC                 JMP VTAB
FB60: 20 58 FC        APPLEII  JSR HOME     ; CLEAR THE SCRN
FB63: A0 08                    LDY #8
FB65: B9 08 FB        STITLE   LDA TITLE-1,Y ; GET A CHAR
FB68: 99 0E 04                 STA LINE1+14,Y
FB6B: 88                       DEY
FB6C: D0 F7                    BNE STITLE
FB6E: 60                       RTS
FB6F: AD F3 03        SETPWRC  LDA SOFTEV+1
FB72: 49 A5                    EOR #$A5
FB74: 8D F4 03                 STA PWREDUP
FB77: 60                       RTS
FB78:                 VIDWAIT  EQU *        ; CHECK FOR A PAUSE
FB78: C9 8D                    CMP #$8D     ; ONLY WHEN I HAVE A CR
FB7A: D0 18                    BNE NOWAIT   ; NOT SO, DO REGULAR
FB7C: AC 00 C0                 LDY KBD      ; IS KEY PRESSED?
FB7F: 10 13                    BPL NOWAIT   ; NO
FB81: C0 93                    CPY #$93     ; IS IT CTL S ?
FB83: D0 0F                    BNE NOWAIT   ; NO SO IGNORE
FB85: 2C 10 C0                 BIT KBDSTRB  ; CLEAR STROBE
FB88: AC 00 C0        KBDWAIT  LDY KBD      ; WAIT TILL NEXT KEY TO RESUME
FB8B: 10 FB                    BPL KBDWAIT  ; WAIT FOR KEYPRESS
FB8D: C0 83                    CPY #$83     ; IS IT CONTROL C ?
FB8F: F0 03                    BEQ NOWAIT   ; YES SO LEAVE IT
FB91: 2C 10 C0                 BIT KBDSTRB  ; CLR STROBE
FB94: 4C FD FB        NOWAIT   JMP VIDOUT   , DO AS BEFORE
FB97:                          PAGE
FB97: 38              ESCOLD   SEC          ; INSURE CARRY SET
FB98: 4C 2C FC                 JMP ESC1
FB9B: A8              ESCNOW   TAY          ; USE CHAR AS INDEX
FB9C: B9 48 FA                 LDA XLTBL-$C9,Y ; XLATE IJKM TO CBAD
FB9F: 20 97 FB                 JSR ESCOLD   ; DO THIS CURSOR MOTION
FBA2: 20 0C FD                 JSR RDKEY    , AND GET NEXT
FBA5: C9 CE           ESCNEW   CMP #$CE     ;IS THIS AN N ?
FBA7: B0 EE                    BCS ESCOLD   ; N OR GREATER DO IT
FBA9: C9 C9                    CMP #$C9     ; LESS THAN I ?
FBAB: 90 EA                    BCC ESCOLD   ; YES SO OLD WAY
FBAD: C9 CC                    CMP #$CC     ; IS IT A L ?
FBAF: F0 E6                    BEQ ESCOLD   ; DO NORMAL
FBB1: D0 E8                    BNE ESCNOW   ; GO DO IT
FBB3: EA                       NOP
FBB4: EA                       NOP
FBB5: EA                       NOP
FBB6: EA                       NOP
FBB7: EA                       NOP
FBB8: EA                       NOP
FBB9: EA                       NOP
FBBA: EA                       NOP
```

```
FBBB:  EA                    NOP
FBBC:  EA                    NOP
FBBD:  EA                    NOP
FBBE:  EA                    NOP
FBBF:  EA                    NOP
FBC0:  EA                    NOP
FBC1:              *     MUST ORG $FBC1
FBC1:  48          BASCALC   PHA
FBC2:  4A                    LSR A
FBC3:  29 03                 AND #$03
FBC5:  09 04                 ORA #$04
FBC7:  85 29                 STA BASH
FBC9:  68                    PLA
FBCA:  29 18                 AND #$18
FBCC:  90 02                 BCC BASCLC2
FBCE:  69 7F                 ADC #$7F
FBD0:  85 28       BASCLC2   STA BASL
FBD2:  0A                    ASL A
FBD3:  0A                    ASL A
FBD4:  05 28                 ORA BASL
FBD6:  85 28                 STA BASL
FBD8:  60                    RTS
FBD9:  C9 87       BELL1     CMP #$87
FBDB:  D0 12                 BNE RTS2B
FBDD:  A9 40                 LDA #$40
FBDF:  20 A8 FC              JSR WAIT
FBE2:  A0 C0                 LDY #$C0
FBE4:  A9 0C       BELL2     LDA #$0C
FBE6:  20 A8 FC              JSR WAIT
FBE9:  AD 30 C0              LDA SPKR
FBEC:  88                    DEY
FBED:  D0 F5                 BNE BELL2
FBEF:  60          RTS2B     RTS
FBF0:                        PAGE
FBF0:  A4 24       STORADV   LDY CH
FBF2:  91 28                 STA (BASL),Y
FBF4:  E6 24       ADVANCE   INC CH
FBF6:  A5 24                 LDA CH
FBF8:  C5 21                 CMP WNDWDTH
FBFA:  B0 66                 BCS CR
FBFC:  60          RTS3      RTS
FBFD:  C9 A0       VIDOUT    CMP #$A0
FBFF:  B0 EF                 BCS STORADV
FC01:  A8                    TAY
FC02:  10 EC                 BPL STORADV
FC04:  C9 8D                 CMP #$8D
FC06:  F0 5A                 BEQ CR
FC08:  C9 8A                 CMP #$8A
FC0A:  F0 5A                 BEQ LF
FC0C:  C9 88                 CMP #$88
FC0E:  D0 C9                 BNE BELL1
FC10:  C6 24       BS        DEC CH
FC12:  10 E8                 BPL RTS3
FC14:  A5 21                 LDA WNDWDTH
FC16:  85 24                 STA CH
FC18:  C6 24                 DEC CH
FC1A:  A5 22       UP        LDA WNDTOP
FC1C:  C5 25                 CMP CV
FC1E:  B0 0B                 BCS RTS4
FC20:  C6 25                 DEC CV
FC22:  A5 25       VTAB      LDA CV
FC24:  20 C1 FB    VTABZ     JSR BASCALC
FC27:  65 20                 ADC WNDLFT
FC29:  85 28                 STA BASL
FC2B:  60          RTS4      RTS
FC2C:  49 C0       ESC1      EOR #$C0     ; ESC @ ?
FC2E:  F0 28                 BEQ HOME     ; IF SO DO HOME AND CLEAR
FC30:  69 FD                 ADC #$FD     ; ESC-A OR B CHECK
FC32:  90 C0                 BCC ADVANCE  ; A, ADVANCE
FC34:  F0 DA                 BEQ BS       ; B, BACKSPACE
FC36:  69 FD                 ADC #$FD     ; ESC-C OR D CHECK
FC38:  90 2C                 BCC LF       ; C, DOWN
FC3A:  F0 DE                 BEQ UP       ; D, GO UP
```

```
FC3C:  69 FD       ADC #$FD     ; ESC-E OR F CKECK
FC3E:  90 5C       BCC CLREOL   ; E, CLEAR TO END OF LINE
FC40:  D0 E9       BNE RTS4     ; ELSE NOT F, RETURN
FC42:  A4 24    CLREOP  LDY CH  ; ESC F IS CLR TO END OF PAGE
FC44:  A5 25            LDA CV
FC46:  48       CLEOP1  PHA
FC47:  20 24 FC         JSR VTABZ
FC4A:  20 9E FC         JSR CLEOLZ
FC4D:  A0 00            LDY #$00
FC4F:  68               PLA
FC50:  69 00            ADC #$00
FC52:  C5 23            CMP WNDBTM
FC54:  90 F0            BCC CLEOP1
FC56:  B0 CA            BCS VTAB
FC58:  A5 22    HOME    LDA WNDTOP
FC5A:  85 25            STA CV
FC5C:  A0 00            LDY #$00
FC5E:  84 24            STY CH
FC60:  F0 E4            BEQ CLEOP1
FC62:                   PAGE
FC62:  A9 00    CR      LDA #$00
FC64:  85 24            STA CH
FC66:  E6 25    LF      INC CV
FC68:  A5 25            LDA CV
FC6A:  C5 23            CMP WNDBTM
FC6C:  90 B6            BCC VTABZ
FC6E:  C6 25            DEC CV
FC70:  A5 22    SCROLL  LDA WNDTOP
FC72:  48               PHA
FC73:  20 24 FC         JSR VTABZ
FC76:  A5 28    SCRL1   LDA BASL
FC78:  85 2A            STA BAS2L
FC7A:  A5 29            LDA BASH
FC7C:  85 2B            STA BAS2H
FC7E:  A4 21            LDY WNDWDTH
FC80:  88               DEY
FC81:  68               PLA
FC82:  69 01            ADC #$01
FC84:  C5 23            CMP WNDBTM
FC86:  B0 0D            BCS SCRL3
FC88:  48               PHA
FC89:  20 24 FC         JSR VTABZ
FC8C:  B1 28    SCRL2   LDA (BASL),Y
FC8E:  91 2A            STA (BAS2L),Y
FC90:  88               DEY
FC91:  10 F9            BPL SCRL2
FC93:  30 E1            BMI SCRL1
FC95:  A0 00    SCRL3   LDY #$00
FC97:  20 9E FC         JSR CLEOLZ
FC9A:  B0 86            BCS VTAB
FC9C:  A4 24    CLREOL  LDY CH
FC9E:  A9 A0    CLEOLZ  LDA #$A0
FCA0:  91 28    CLEOL2  STA (BASL),Y
FCA2:  C8               INY
FCA3:  C4 21            CPY WNDWDTH
FCA5:  90 F9            BCC CLEOL2
FCA7:  60               RTS
FCA8:  38       WAIT    SEC
FCA9:  48       WAIT2   PHA
FCAA:  E9 01    WAIT3   SBC #$01
FCAC:  D0 FC            BNE WAIT3
FCAE:  68               PLA
FCAF:  E9 01            SBC #$01
FCB1:  D0 F6            BNE WAIT2
FCB3:  60               RTS
FCB4:  E6 42    NXTA4   INC A4L
FCB6:  D0 02            BNE NXTA1
FCB8:  E6 43            INC A4H
FCBA:  A5 3C    NXTA1   LDA A1L
FCBC:  C5 3E            CMP A2L
FCBE:  A5 3D            LDA A1H
FCC0:  E5 3F            SBC A2H
FCC2:  E6 3C            INC A1L
```

```
FCC4:  D0 02                 BNE RTS4B
FCC6:  E6 3D                 INC A1H
FCC8:  60          RTS4B     RTS
FCC9:                        PAGE
FCC9:  A0 4B       HEADR     LDY #$4B
FCCB:  20 DB FC              JSR ZERDLY
FCCE:  D0 F9                 BNE HEADR
FCD0:  69 FE                 ADC #$FE
FCD2:  B0 F5                 BCS HEADR
FCD4:  A0 21                 LDY #$21
FCD6:  20 DB FC    WRBIT     JSR ZERDLY
FCD9:  C8                    INY
FCDA:  C8                    INY
FCDB:  88          ZERDLY    DEY
FCDC:  D0 FD                 BNE ZERDLY
FCDE:  90 05                 BCC WRTAPE
FCE0:  A0 32                 LDY #$32
FCE2:  88          ONEDLY    DEY
FCE3:  D0 FD                 BNE ONEDLY
FCE5:  AC 20 C0    WRTAPE    LDY TAPEOUT
FCE8:  A0 2C                 LDY #$2C
FCEA:  CA                    DEX
FCEB:  60                    RTS
FCEC:  A2 08       RDBYTE    LDX #$08
FCEE:  48          RDBYT2    PHA
FCEF:  20 FA FC              JSR RD2BIT
FCF2:  68                    PLA
FCF3:  2A                    ROL A
FCF4:  A0 3A                 LDY #$3A
FCF6:  CA                    DEX
FCF7:  D0 F5                 BNE RDBYT2
FCF9:  60                    RTS
FCFA:  20 FD FC    RD2BIT    JSR RDBIT
FCFD:  88          RDBIT     DEY
FCFE:  AD 60 C0              LDA TAPEIN
FD01:  45 2F                 EOR LASTIN
FD03:  10 F8                 BPL RDBIT
FD05:  45 2F                 EOR LASTIN
FD07:  85 2F                 STA LASTIN
FD09:  C0 80                 CPY #$80
FD0B:  60                    RTS
FD0C:  A4 24       RDKEY     LDY CH
FD0E:  B1 28                 LDA (BASL),Y
FD10:  48                    PHA
FD11:  29 3F                 AND #$3F
FD13:  09 40                 ORA #$40
FD15:  91 28                 STA (BASL),Y
FD17:  68                    PLA
FD18:  6C 38 00              JMP (KSWL)
FD1B:  E6 4E       KEYIN     INC RNDL
FD1D:  D0 02                 BNE KEYIN2
FD1F:  E6 4F                 INC RNDH
FD21:  2C 00 C0    KEYIN2    BIT KBD       ; READ KEYBOARD
FD24:  10 F5                 BPL KEYIN
FD26:  91 28                 STA (BASL),Y
FD28:  AD 00 C0              LDA KBD
FD2B:  2C 10 C0              BIT KBDSTRB
FD2E:  60                    RTS
FD2F:  20 0C FD    ESC       JSR RDKEY
FD32:  20 A5 FB              JSR ESCNEW
FD35:  20 0C FD    RDCHAR    JSR RDKEY
FD38:  C9 9B                 CMP #$9B
FD3A:  F0 F3                 BEQ ESC
FD3C:  60                    RTS
FD3D:                        PAGE
FD3D:  A5 32       NOTCR     LDA INVFLG
FD3F:  48                    PHA
FD40:  A9 FF                 LDA #$FF
FD42:  85 32                 STA INVFLG
FD44:  BD 00 02              LDA IN,X
FD47:  20 ED FD              JSR COUT
FD4A:  68                    PLA
FD4B:  85 32                 STA INVFLG
```

```
FD4D:  BD 00 02                LDA IN,X
FD50:  C9 88                   CMP #$88
FD52:  F0 1D                   BEQ BCKSPC
FD54:  C9 98                   CMP #$98
FD56:  F0 0A                   BEQ CANCEL
FD58:  E0 F8                   CPX #$F8
FD5A:  90 03                   BCC NOTCR1
FD5C:  20 3A FF                JSR BELL
FD5F:  E8             NOTCR1   INX
FD60:  D0 13                   BNE NXTCHAR
FD62:  A9 DC          CANCEL   LDA #$DC
FD64:  20 ED FD                JSR COUT
FD67:  20 8E FD       GETLNZ   JSR CROUT
FD6A:  A5 33          GETLN    LDA PROMPT
FD6C:  20 ED FD                JSR COUT
FD6F:  A2 01                   LDX #$01
FD71:  8A             BCKSPC   TXA
FD72:  F0 F3                   BEQ GETLNZ
FD74:  CA                      DEX
FD75:  20 35 FD       NXTCHAR  JSR RDCHAR
FD78:  C9 95                   CMP #$95
FD7A:  D0 02                   BNE CAPTST
FD7C:  B1 28                   LDA (BASL),Y
FD7E:  C9 E0          CAPTST   CMP #$E0
FD80:  90 02                   BCC ADDINP
FD82:  29 DF                   AND #$DF     ; SHIFT TO UPPER CASE
FD84:  9D 00 02       ADDINP   STA IN,X
FD87:  C9 8D                   CMP #$8D
FD89:  D0 B2                   BNE NOTCR
FD8B:  20 9C FC                JSR CLREOL
FD8E:  A9 8D          CROUT    LDA #$8D
FD90:  D0 5B                   BNE COUT
FD92:  A4 3D          PRA1     LDY A1H
FD94:  A6 3C                   LDX A1L
FD96:  20 8E FD       PRYX2    JSR CROUT
FD99:  20 40 F9                JSR PRNTYX
FD9C:  A0 00                   LDY #$00
FD9E:  A9 AD                   LDA #$AD
FDA0:  4C ED FD                JMP COUT
FDA3                           PAGE
FDA3   A5 3C          XAM8     LDA A1L
FDA5:  09 07                   ORA #$07
FDA7:  85 3E                   STA A2L
FDA9:  A5 3D                   LDA A1H
FDAB:  85 3F                   STA A2H
FDAD:  A5 3C          MOD8CHK  LDA A1L
FDAF:  29 07                   AND #$07
FDB1:  D0 03                   BNE DATAOUT
FDB3:  20 92 FD       XAM      JSR PRA1
FDB6:  A9 A0          DATAOUT  LDA #$A0
FDB8:  20 ED FD                JSR COUT
FDBB:  B1 3C                   LDA (A1L),Y
FDBD:  20 DA FD                JSR PRBYTE
FDC0:  20 BA FC                JSR NXTA1
FDC3:  90 E8                   BCC MOD8CHK
FDC5:  60             RTS4C    RTS
FDC6:  4A             XAMPM    LSR A
FDC7:  90 EA                   BCC XAM
FDC9:  4A                      LSR A
FDCA:  4A                      LSR A
FDCB:  A5 3E                   LDA A2L
FDCD:  90 02                   BCC ADD
FDCF:  49 FF                   EOR #$FF
FDD1:  65 3C          ADD      ADC A1L
FDD3:  48                      PHA
FDD4:  A9 BD                   LDA #$BD
FDD6:  20 ED FD                JSR COUT
FDD9:  68                      PLA
FDDA:  48             PRBYTE   PHA
FDDB:  4A                      LSR A
FDDC:  4A                      LSR A
FDDD:  4A                      LSR A
FDDE:  4A                      LSR A
```

```
FDDF:  20 E5 FD            JSR PRHEXZ
FDE2:  68                  PLA
FDE3:  29 0F      PRHEX    AND #$0F
FDE5:  09 B0      PRHEXZ   ORA #$B0
FDE7:  C9 BA               CMP #$BA
FDE9:  90 02               BCC COUT
FDEB:  69 06               ADC #$06
FDED:  6C 36 00   COUT     JMP (CSWL)
FDF0:  C9 A0      COUT1    CMP #$A0
FDF2:  90 02               BCC COUTZ
FDF4:  25 32               AND INVFLG
FDF6:  84 35      COUTZ    STY YSAV1
FDF8:  48                  PHA
FDF9:  20 78 FB            JSR VIDWAIT ; GO CHECK FOR PAUSE
FDFC:  68                  PLA
FDFD:  A4 35               LDY YSAV1
FDFF:  60                  RTS
FE00:                      PAGE
FE00:  C6 34      BL1      DEC YSAV
FE02:  F0 9F               BEQ XAM8
FE04:  CA         BLANK    DEX
FE05:  D0 16               BNE SETMDZ
FE07:  C9 BA               CMP #$BA
FE09:  D0 BB               BNE XAMPM
FE0B:  85 31      STOR     STA MODE
FE0D:  A5 3E               LDA A2L
FE0F:  91 40               STA (A3L),Y
FE11:  E6 40               INC A3L
FE13:  D0 02               BNE RTS5
FE15:  E6 41               INC A3H
FE17:  60         RTS5     RTS
FE18:  A4 34      SETMODE  LDY YSAV
FE1A:  B9 FF 01            LDA IN-1,Y
FE1D:  85 31      SETMDZ   STA MODE
FE1F:  60                  RTS
FE20:  A2 01      LT       LDX #$01
FE22:  B5 3E      LT2      LDA A2L,X
FE24:  95 42               STA A4L,X
FE26:  95 44               STA A5L,X
FE28:  CA                  DEX
FE29:  10 F7               BPL LT2
FE2B:  60                  RTS
FE2C:  B1 3C      MOVE     LDA (A1L),Y
FE2E:  91 42               STA (A4L),Y
FE30:  20 B4 FC            JSR NXTA4
FE33:  90 F7               BCC MOVE
FE35:  60                  RTS
FE36:  B1 3C      VFY      LDA (A1L),Y
FE38:  D1 42               CMP (A4L),Y
FE3A:  F0 1C               BEQ VFYOK
FE3C:  20 92 FD            JSR PRA1
FE3F:  B1 3C               LDA (A1L),Y
FE41:  20 DA FD            JSR PRBYTE
FE44:  A9 A0               LDA #$A0
FE46:  20 ED FD            JSR COUT
FE49:  A9 A8               LDA #$A8
FE4B:  20 ED FD            JSR COUT
FE4E:  B1 42               LDA (A4L),Y
FE50:  20 DA FD            JSR PRBYTE
FE53:  A9 A9               LDA #$A9
FE55:  20 ED FD            JSR COUT
FE58:  20 B4 FC   VFYOK    JSR NXTA4
FE5B:  90 D9               BCC VFY
FE5D:  60                  RTS
FE5E:  20 75 FE   LIST     JSR A1PC
FE61:  A9 14               LDA #$14
FE63:  48         LIST2    PHA
FE64:  20 D0 F8            JSR INSTDSP
FE67:  20 53 F9            JSR PCADJ
FE6A:  85 3A               STA PCL
FE6C:  84 3B               STY PCH
FE6E:  68                  PLA
FE6F:  38                  SEC
```

```
FE70:  E9 01                  SBC #$01
FE72:  D0 EF                  BNE LIST2
FE74:  60                     RTS
FE75:                         PAGE
FE75:  8A          A1PC       TXA
FE76:  F0 07                  BEQ A1PCRTS
FE78:  B5 3C       A1PCLP     LDA A1L,X
FE7A:  95 3A                  STA PCL,X
FE7C:  CA                     DEX
FE7D:  10 F9                  BPL A1PCLP
FE7F:  60          A1PCRTS    RTS
FE80:  A0 3F       SETINV     LDY #$3F
FE82:  D0 02                  BNE SETIFLG
FE84:  A0 FF       SETNORM    LDY #$FF
FE86:  84 32       SETIFLG    STY INVFLG
FE88:  60                     RTS
FE89:  A9 00       SETKBD     LDA #$00
FE8B:  85 3E       INPORT     STA A2L
FE8D:  A2 38       INPRT      LDX #KSWL
FE8F:  A0 1B                  LDY #KEYIN
FE91:  D0 08                  BNE IOPRT
FE93:  A9 00       SETVID     LDA #$00
FE95:  85 3E       OUTPORT    STA A2L
FE97:  A2 36       OUTPRT     LDX #CSWL
FE99:  A0 F0                  LDY #COUT1
FE9B:  A5 3E       IOPRT      LDA A2L
FE9D:  29 0F                  AND #$0F
FE9F:  F0 06                  BEQ IOPRT1
FEA1:  09 C0                  ORA #IOADR/256
FEA3:  A0 00                  LDY #$00
FEA5:  F0 02                  BEQ IOPRT2
FEA7:  A9 FD       IOPRT1     LDA #COUT1/256
FEA9:              IOPRT2     EQU *
FEA9:  94 00                  STY LOC0,X ;  $94,$00
FEAB:  95 01                  STA LOC1,X ;  $95,$01
FEAD:  60                     RTS
FEAE:  EA                     NOP
FEAF:  EA                     NOP
FEB0:  4C 00 E0    XBASIC     JMP BASIC
FEB3:  4C 03 E0    BASCONT    JMP BASIC2
FEB6:  20 75 FE    GO         JSR A1PC
FEB9:  20 3F FF               JSR RESTORE
FEBC:  6C 3A 00               JMP (PCL)
FEBF:  4C D7 FA    REGZ       JMP REGDSP
FEC2:  60          TRACE      RTS
FEC3:              * TRACE IS GONE
FEC3:  EA                     NOP
FEC4:  60          STEPZ      RTS          ; STEP IS GONE
FEC5:  EA                     NOP
FEC6:  EA                     NOP
FEC7:  EA                     NOP
FEC8:  EA                     NOP
FEC9:  EA                     NOP
FECA:  4C F8 03    USR        JMP USRADR
FECD:                         PAGE
FECD:  A9 40       WRITE      LDA #$40
FECF:  20 C9 FC               JSR HEADR
FED2:  A0 27                  LDY #$27
FED4:  A2 00       WR1        LDX #$00
FED6:  41 3C                  EOR (A1L,X)
FED8:  48                     PHA
FED9:  A1 3C                  LDA (A1L,X)
FEDB:  20 ED FE               JSR WRBYTE
FEDE:  20 BA FC               JSR NXTA1
FEE1:  A0 1D                  LDY #$1D
FEE3:  68                     PLA
FEE4:  90 EE                  BCC WR1
FEE6:  A0 22                  LDY #$22
FEE8:  20 ED FE               JSR WRBYTE
FEEB:  F0 4D                  BEQ BELL
FEED:  A2 10       WRBYTE     LDX #$10
FEEF:  0A          WRBYT2     ASL A
FEF0:  20 D6 FC               JSR WRBIT
```

```
FEF3:  D0 FA              BNE WRBYT2
FEF5:  60                 RTS
FEF6:  20 00 FE  CRMON    JSR BL1
FEF9:  68                 PLA
FEFA:  68                 PLA
FEFB:  D0 6C              BNE MONZ
FEFD:  20 FA FC  READ     JSR RD2BIT
FF00:  A9 16              LDA #$16
FF02:  20 C9 FC           JSR HEADR
FF05:  85 2E              STA CHKSUM
FF07:  20 FA FC           JSR RD2BIT
FF0A:  A0 24     RD2      LDY #$24
FF0C:  20 FD FC           JSR RDBIT
FF0F:  B0 F9              BCS RD2
FF11:  20 FD FC           JSR RDBIT
FF14:  A0 3B              LDY #$3B
FF16:  20 EC FC  RD3      JSR RDBYTE
FF19:  81 3C              STA (A1L,X)
FF1B:  45 2E              EOR CHKSUM
FF1D:  85 2E              STA CHKSUM
FF1F:  20 BA FC           JSR NXTA1
FF22:  A0 35              LDY #$35
FF24:  90 F0              BCC RD3
FF26:  20 EC FC           JSR RDBYTE
FF29:  C5 2E              CMP CHKSUM
FF2B:  F0 0D              BEQ BELL
FF2D:  A9 C5     PRERR    LDA #$C5
FF2F:  20 ED FD           JSR COUT
FF32:  A9 D2              LDA #$D2
FF34:  20 ED FD           JSR COUT
FF37:  20 ED FD           JSR COUT
FF3A:  A9 87     BELL     LDA #$87
FF3C:  4C ED FD           JMP COUT
FF3F:                     PAGE
FF3F:  A5 48     RESTORE  LDA STATUS
FF41:  48                 PHA
FF42:  A5 45              LDA A5H
FF44:  A6 46     RESTR1   LDX XREG
FF46:  A4 47              LDY YREG
FF48:  28                 PLP
FF49:  60                 RTS
FF4A:  85 45     SAVE     STA A5H
FF4C:  86 46     SAV1     STX XREG
FF4E:  84 47              STY YREG
FF50:  08                 PHP
FF51:  68                 PLA
FF52:  85 48              STA STATUS
FF54:  BA                 TSX
FF55:  86 49              STX SPNT
FF57:  D8                 CLD
FF58:  60                 RTS
FF59:  20 84 FE  OLDRST   JSR SETNORM
FF5C:  20 2F FB           JSR INIT
FF5F:  20 93 FE           JSR SETVID
FF62:  20 89 FE           JSR SETKBD
FF65:                     PAGE
FF65:  D8        MON      CLD
FF66:  20 3A FF           JSR BELL
FF69:  A9 AA     MONZ     LDA #$AA
FF6B:  85 33              STA PROMPT
FF6D:  20 67 FD           JSR GETLNZ
FF70:  20 C7 FF           JSR ZMODE
FF73:  20 A7 FF  NXTITM   JSR GETNUM
FF76:  84 34              STY YSAV
FF78:  A0 17              LDY #$17
FF7A:  88        CHRSRCH  DEY
FF7B:  30 E8              BMI MON
FF7D:  D9 CC FF           CMP CHRTBL,Y
FF80:  D0 F8              BNE CHRSRCH
FF82:  20 BE FF           JSR TOSUB
FF85:  A4 34              LDY YSAV
FF87:  4C 73 FF           JMP NXTITM
FF8A:  A2 03     DIG      LDX #$03
```

```
FF8C:  0A                          ASL  A
FF8D:  0A                          ASL  A
FF8E:  0A                          ASL  A
FF8F:  0A                          ASL  A
FF90:  0A           NXTBIT         ASL  A
FF91:  26 3E                       ROL  A2L
FF93:  26 3F                       ROL  A2H
FF95:  CA                          DEX
FF96:  10 F8                       BPL  NXTBIT
FF98:  A5 31        NXTBAS         LDA  MODE
FF9A:  D0 06                       BNE  NXTBS2
FF9C:               *
FF9C:  B5 3F                       LDA  A2H,X
FF9E:               *
FF9E:  95 3D                       STA  A1H,X
FFA0:               *
FFA0:  95 41                       STA  A3H,X
FFA2:  E8           NXTBS2         INX
FFA3:  F0 F3                       BEQ  NXTBAS
FFA5:  D0 06                       BNE  NXTCHR
FFA7:  A2 00        GETNUM         LDX  #$00
FFA9:  86 3E                       STX  A2L
FFAB:  86 3F                       STX  A2H
FFAD:  B9 00 02     NXTCHR         LDA  IN,Y
FFB0:  C8                          INY
FFB1:  49 B0                       EOR  #$B0
FFB3:  C9 0A                       CMP  #$0A
FFB5:  90 D3                       BCC  DIG
FFB7:  69 88                       ADC  #$88
FFB9:  C9 FA                       CMP  #$FA
FFBB:  B0 CD                       BCS  DIG
FFBD:  60                          RTS
FFBE:  A9 FE        TOSUB          LDA  #GO/256
FFC0:  48                          PHA
FFC1:  B9 E3 FF                    LDA  SUBTBL,Y
FFC4:  48                          PHA
FFC5:  A5 31                       LDA  MODE
FFC7:  A0 00        ZMODE          LDY  #$00
FFC9:  84 31                       STY  MODE
FFCB:  60                          RTS
FFCC:                              PAGE
FFCC:  BC           CHRTBL         DFB  $BC
FFCD:  B2                          DFB  $B2
FFCE:  BE                          DFB  $BE
FFCF:  B2                          DFB  $B2      ; T CMD NOW LIKE USR
FFD0:  EF                          DFB  $EF
FFD1:  C4                          DFB  $C4
FFD2:  B2                          DFB  $B2      ; S CMD NOW LIKE USR
FFD3:  A9                          DFB  $A9
FFD4:  BB                          DFB  $BB
FFD5:  A6                          DFB  $A6
FFD6:  A4                          DFB  $A4
FFD7:  06                          DFB  $06
FFD8:  95                          DFB  $95
FFD9:  07                          DFB  $07
FFDA:  02                          DFB  $02
FFDB:  05                          DFB  $05
FFDC:  F0                          DFB  $F0
FFDD:  00                          DFB  $00
FFDE:  EB                          DFB  $EB
FFDF:  93                          DFB  $93
FFE0:  A7                          DFB  $A7
FFE1:  C6                          DFB  $C6
FFE2:  99                          DFB  $99
FFE3:  B2           SUBTBL         DFB  $B2
FFE4:  C9                          DFB  $C9
FFE5:  BE                          DFB  $BE
FFE6:  C1                          DFB  $C1
FFE7:  35                          DFB  $35
FFE8:  8C                          DFB  $8C
FFE9:  C4                          DFB  $C4
FFEA:  96                          DFB  $96
FFEB:  AF                          DFB  $AF
```

```
FFEC:  17                 DFB $17
FFED:  17                 DFB $17
FFEE:  2B                 DFB $2B
FFEF:  1F                 DFB $1F
FFF0:  83                 DFB $B3
FFF1:  7F                 DFB $7F
FFF2:  5D                 DFB $5D
FFF3:  CC                 DFB $CC
FFF4:  B5                 DFB $B5
FFF5:  FC                 DFB $FC
FFF6:  17                 DFB $17
FFF7:  17                 DFB $17
FFF8:  F5                 DFB $F5
FFF9:  03                 DFB $03
FFFA:  FB 03              DW  NMI
FFFC:  62 FA              DW  RESET
FFFE:  40 FA              DW  IRQ

ENDASM
```

MONITOR ROM LISTING

```
*****************************
*                           *
*         APPLE II          *
*      SYSTEM MONITOR       *
*                           *
*    COPYRIGHT 1977 BY      *
*   APPLE COMPUTER, INC.    *
*                           *
*   ALL RIGHTS RESERVED     *
*                           *
*        S. WOZNIAK         *
*         A. BAUM           *
*                           *
*****************************
         TITLE                   "APPLE II SYSTEM MONITOR"
LOC0       EPZ   $00
LOC1       EPZ   $01
WNDLFT     EPZ   $20
WNDWDTH    EPZ   $21
WNDTOP     EPZ   $22
WNDBTM     EPZ   $23
CH         EPZ   $24
CV         EPZ   $25
GBASL      EPZ   $26
GBASH      EPZ   $27
BASL       EPZ   $28
BASH       EPZ   $29
BAS2L      EPZ   $2A
BAS2H      EPZ   $2B
H2         EPZ   $2C
LMNEM      EPZ   $2C
RTNL       EPZ   $2C
V2         EPZ   $2D
RMNEM      EPZ   $2D
RTNH       EPZ   $2D
MASK       EPZ   $2E
CHKSUM     EPZ   $2E
FORMAT     EPZ   $2E
LASTIN     EPZ   $2F
LENGTH     EPZ   $2F
SIGN       EPZ   $2F
COLOR      EPZ   $30
MODE       EPZ   $31
INVFLG     EPZ   $32
PROMPT     EPZ   $33
YSAV       EPZ   $34
YSAV1      EPZ   $35
CSWL       EPZ   $36
CSWH       EPZ   $37
KSWL       EPZ   $38
KSWH       EPZ   $39
PCL        EPZ   $3A
PCH        EPZ   $3B
XQT        EPZ   $3C
A1L        EPZ   $3C
A1H        EPZ   $3D
A2L        EPZ   $3E
A2H        EPZ   $3F
A3L        EPZ   $40
A3H        EPZ   $41
A4L        EPZ   $42
A4H        EPZ   $43
A5L        EPZ   $44
A5H        EPZ   $45
ACC        EPZ   $45
XREG       EPZ   $46
YREG       EPZ   $47
STATUS     EPZ   $48
```

```
                SPNT     EPZ  $49
                RNDL     EPZ  $4E
                RNDH     EPZ  $4F
                ACL      EPZ  $50
                ACH      EPZ  $51
                XTNDL    EPZ  $52
                XTNDH    EPZ  $53
                AUXL     EPZ  $54
                AUXH     EPZ  $55
                PICK     EPZ  $95
                IN       EQU  $0200
                USRADR   EQU  $03F8
                NMI      EQU  $03FB
                IRQLOC   EQU  $03FE
                IOADR    EQU  $C000
                KBD      EQU  $C000
                KBDSTRB  EQU  $C010
                TAPEOUT  EQU  $C020
                SPKR     EQU  $C030
                TXTCLR   EQU  $C050
                TXTSET   EQU  $C051
                MIXCLR   EQU  $C052
                MIXSET   EQU  $C053
                LOWSCR   EQU  $C054
                HISCR    EQU  $C055
                LORES    EQU  $C056
                HIRES    EQU  $C057
                TAPEIN   EQU  $C060
                PADDL0   EQU  $C064
                PTRIG    EQU  $C070
                BASIC    EQU  $E000
                BASIC2   EQU  $E003
                         ORG  $F800        ROM START ADDRESS
F800:  4A       PLOT     LSR  A            Y-COORD/2
F801:  08                PHP               SAVE LSB IN CARRY
F802:  20 47 F8          JSR  GBASCALC     CALC BASE ADR IN GBASL,H
F805:  28                PLP               RESTORE LSB FROM CARRY
F806:  A9 0F             LDA  #$0F         MASK $0F IF EVEN
F808:  90 02             BCC  RTMASK
F80A:  69 E0             ADC  #$E0         MASK $F0 IF ODD
F80C:  85 2E    RTMASK   STA  MASK
F80E:  B1 26    PLOT1    LDA  (GBASL),Y    DATA
F810:  45 30             EOR  COLOR         XOR  COLOR
F812:  25 2E             AND  MASK           AND MASK
F814:  51 26             EOR  (GBASL),Y        XOR DATA
F816:  91 26             STA  (GBASL),Y         TO DATA
F818:  60                RTS
F819:  20 00 F8 HLINE    JSR  PLOT         PLOT SQUARE
F81C:  C4 2C    HLINE1   CPY  H2           DONE?
F81E:  B0 11             BCS  RTS1          YES, RETURN
F820:  C8                INY                NO, INCR INDEX (X-COORD)
F821:  20 0E F8          JSR  PLOT1        PLOT NEXT SQUARE
F824:  90 F6             BCC  HLINE1       ALWAYS TAKEN
F826:  69 01    VLINEZ   ADC  #$01         NEXT Y-COORD
F828:  48       VLINE    PHA                SAVE ON STACK
F829:  20 00 F8          JSR  PLOT          PLOT SQUARE
F82C:  68                PLA
F82D:  C5 2D             CMP  V2           DONE?
F82F:  90 F5             BCC  VLINEZ        NO,LOOP.
F831:  60       RTS1     RTS
F832:  A0 2F    CLRSCR   LDY  #$2F         MAX Y, FULL SCRN CLR
F834:  D0 02             BNE  CLRSC2       ALWAYS TAKEN
F836:  A0 27    CLRTOP   LDY  #$27         MAX Y, TOP SCRN CLR
F838:  84 2D    CLRSC2   STY  V2           STORE AS BOTTOM COORD
                *                        FOR VLINE CALLS
F83A:  A0 27             LDY  #$27         RIGHTMOST X-COORD (COLUMN)
F83C:  A9 00    CLRSC3   LDA  #$0          TOP COORD FOR VLINE CALLS
F83E:  85 30             STA  COLOR        CLEAR COLOR (BLACK)
F840:  20 28 F8          JSR  VLINE        DRAW VLINE
F843:  88                DEY               NEXT LEFTMOST X-COORD
F844:  10 F6             BPL  CLRSC3       LOOP UNTIL DONE.
F846:  60                RTS
F847:  48       GBASCALC PHA               FOR INPUT 000DEFGH
F848:  4A                LSR  A
```

```
F849:  29 03              AND  #$03
F84B:  09 04              ORA  #$04           GENERATE GBASH=000001FG
F84D:  85 27              STA  GBASH
F84F:  68                 PLA                 AND GBASL=HDEDE000
F850:  29 18              AND  #$18
F852:  90 02              BCC  GBCALC
F854:  69 7F              ADC  #$7F
F856:  85 26     GBCALC   STA  GBASL
F858:  0A                 ASL  A
F859:  0A                 ASL  A
F85A:  05 26              ORA  GBASL
F85C:  85 26              STA  GBASL
F85E:  60                 RTS
F85F:  A5 30     NXTCOL   LDA  COLOR          INCREMENT COLOR BY 3
F861:  18                 CLC
F862:  69 03              ADC  #$03
F864:  29 0F     SETCOL   AND  #$0F           SETS COLOR=17*A MOD 16
F866:  85 30              STA  COLOR
F868:  0A                 ASL  A              BOTH HALF BYTES OF COLOR EQUAL
F869:  0A                 ASL  A
F86A:  0A                 ASL  A
F86B:  0A                 ASL  A
F86C:  05 30              ORA  COLOR
F86E:  85 30              STA  COLOR
F870:  60                 RTS
F871:  4A        SCRN     LSR  A              READ SCREEN Y-COORD/2
F872:  08                 PHP                 SAVE LSB (CARRY)
F873:  20 47 F8           JSR  GBASCALC       CALC BASE ADDRESS
F876:  B1 26              LDA  (GBASL),Y      GET BYTE
F878:  28                 PLP                 RESTORE LSB FROM CARRY
F879:  90 04     SCRN2    BCC  RTMSKZ         IF EVEN, USE LO H
F87B:  4A                 LSR  A
F87C:  4A                 LSR  A
F87D:  4A                 LSR  A              SHIFT HIGH HALF BYTE DOWN
F87E:  4A                 LSR  A
F87F:  29 0F     RTMSKZ   AND  #$0F           MASK 4-BITS
F881:  60                 RTS
F882:  A6 3A     INSDS1   LDX  PCL            PRINT PCL,H
F884:  A4 3B              LDY  PCH
F886:  20 96 FD           JSR  PRYX2
F889:  20 48 F9           JSR  PRBLNK         FOLLOWED BY A BLANK
F88C:  A1 3A              LDA  (PCL,X)        GET OP CODE
F88E:  A8        INSDS2   TAY
F88F:  4A                 LSR  A              EVEN/ODD TEST
F890:  90 09              BCC  IEVEN
F892:  6A                 ROR  A              BIT 1 TEST
F893:  B0 10              BCS  ERR            XXXXXX11 INVALID OP
F895:  C9 A2              CMP  #$A2
F897:  F0 0C              BEQ  ERR            OPCODE $89 INVALID
F899:  29 87              AND  #$87           MASK BITS
F89B:  4A        IEVEN    LSR  A              LSB INTO CARRY FOR L/R TEST
F89C:  AA                 TAX
F89D:  BD 62 F9           LDA  FMT1,X         GET FORMAT INDEX BYTE
F8A0:  20 79 F8           JSR  SCRN2          R/L H-BYTE ON CARRY
F8A3:  D0 04              BNE  GETFMT
F8A5:  A0 80     ERR      LDY  #$80           SUBSTITUTE $80 FOR INVALID OPS
F8A7:  A9 00              LDA  #$0            SET PRINT FORMAT INDEX TO 0
F8A9:  AA        GETFMT   TAX
F8AA:  BD A6 F9           LDA  FMT2,X         INDEX INTO PRINT FORMAT TABLE
F8AD:  85 2E              STA  FORMAT         SAVE FOR ADR FIELD FORMATTING
F8AF:  29 03              AND  #$03           MASK FOR 2-BIT LENGTH
                 *          (P=1 BYTE, 1=2 BYTE, 2=3 BYTE)
F8B1:  85 2F              STA  LENGTH
F8B3:  98                 TYA                 OPCODE
F8B4:  29 8F              AND  #$8F           MASK FOR 1XXX1010 TEST
F8B6:  AA                 TAX                  SAVE IT
F8B7:  98                 TYA                 OPCODE TO A AGAIN
F8B8:  A0 03              LDY  #$03
F8BA:  E0 8A              CPX  #$8A
F8BC:  F0 0B              BEQ  MNNDX3
F8BE:  4A        MNNDX1   LSR  A
F8BF:  90 08              BCC  MNNDX3         FORM INDEX INTO MNEMONIC TABLE
F8C1:  4A                 LSR  A
```

```
F8C2:  4A              MNNDX2   LSR  A              1) 1XXX1010=>00101XXX
F8C3:  09 20                    ORA  #$20           2) XXXYYY01=>00111XXX
F8C5:  88                       DEY                  3) XXXYYY10=>00110XXX
F8C6:  D0 FA                    BNE  MNNDX2         4) XXXYY100=>00100XXX
F8C8:  C8                       INY                  5) XXXXX000=>000XXXXX
F8C9:  88              MNNDX3   DEY
F8CA:  D0 F2                    BNE  MNNDX1
F8CC:  60                       RTS
F8CD:  FF FF FF                 DFB  $FF,$FF,$FF
F8D0:  20 82 F8        INSTDSP  JSR  INSDS1         GEN FMT, LEN BYTES
F8D3:  48                       PHA                 SAVE MNEMONIC TABLE INDEX
F8D4:  B1 3A           PRNTOP   LDA  (PCL),Y
F8D6:  20 DA FD                 JSR  PRBYTE
F8D9:  A2 01                    LDX  #$01           PRINT 2 BLANKS
F8DB:  20 4A F9        PRNTBL   JSR  PRBL2
F8DE:  C4 2F                    CPY  LENGTH         PRINT INST (1-3 BYTES)
F8E0:  C8                       INY                 IN A 12 CHR FIELD
F8E1:  90 F1                    BCC  PRNTOP
F8E3:  A2 03                    LDX  #$03           CHAR COUNT FOR MNEMONIC PRINT
F8E5:  C0 04                    CPY  #$04
F8E7:  90 F2                    BCC  PRNTBL
F8E9:  68                       PLA                 RECOVER MNEMONIC INDEX
F8EA:  A8                       TAY
F8EB:  B9 C0 F9                 LDA  MNEML,Y
F8EE:  85 2C                    STA  LMNEM          FETCH 3-CHAR MNEMONIC
F8F0:  B9 00 FA                 LDA  MNEMR,Y          (PACKED IN 2-BYTES)
F8F3:  85 2D                    STA  RMNEM
F8F5:  A9 00           PRMN1    LDA  #$00
F8F7:  A0 05                    LDY  #$05
F8F9:  06 2D           PRMN2    ASL  RMNEM          SHIFT 5 BITS OF
F8FB:  26 2C                    ROL  LMNEM            CHARACTER INTO A
F8FD:  2A                       ROL  A                  (CLEARS CARRY)
F8FE:  88                       DEY
F8FF:  D0 F8                    BNE  PRMN2
F901:  69 BF                    ADC  #$BF           ADD "?" OFFSET
F903:  20 ED FD                 JSR  COUT           OUTPUT A CHAR OF MNEM
F906:  CA                       DEX
F907:  D0 EC                    BNE  PRMN1
F909:  20 48 F9                 JSR  PRBLNK         OUTPUT 3 BLANKS
F90C:  A4 2F                    LDY  LENGTH
F90E:  A2 06                    LDX  #$06           CNT FOR 6 FORMAT BITS
F910:  E0 03           PRADR1   CPX  #$03
F912:  F0 1C                    BEQ  PRADR5         IF X=3 THEN ADDR.
F914:  06 2E           PRADR2   ASL  FORMAT
F916:  90 0E                    BCC  PRADR3
F918:  BD B3 F9                 LDA  CHAR1-1,X
F91B:  20 ED FD                 JSR  COUT
F91E:  BD B9 F9                 LDA  CHAR2-1,X
F921:  F0 03                    BEQ  PRADR3
F923:  20 ED FD                 JSR  COUT
F926:  CA              PRADR3   DEX
F927:  D0 E7                    BNE  PRADR1
F929:  60                       RTS
F92A:  88              PRADR4   DEY
F92B:  30 E7                    BMI  PRADR2
F92D:  20 DA FD                 JSR  PRBYTE
F930:  A5 2E           PRADR5   LDA  FORMAT
F932:  C9 E8                    CMP  #$E8           HANDLE REL ADR MODE
F934:  B1 3A                    LDA  (PCL),Y        SPECIAL (PRINT TARGET,
F936:  90 F2                    BCC  PRADR4           NOT OFFSET)
F938:  20 56 F9        RELADR   JSR  PCADJ3
F93B:  AA                       TAX                 PCL,PCH+OFFSET+1 TO A,Y
F93C:  E8                       INX
F93D:  D0 01                    BNE  PRNTYX         +1 TO Y,X
F93F:  C8                       INY
F940:  98              PRNTYX   TYA
F941:  20 DA FD        PRNTAX   JSR  PRBYTE         OUTPUT TARGET ADR
F944:  8A              PRNTX    TXA                   OF BRANCH AND RETURN
F945:  4C DA FD                 JMP  PRBYTE
F948:  A2 03           PRBLNK   LDX  #$03           BLANK COUNT
F94A:  A9 A0           PRBL2    LDA  #$A0           LOAD A SPACE
F94C:  20 ED FD        PRBL3    JSR  COUT           OUTPUT A BLANK
F94F:  CA                       DEX
```

```
F950:  D0 F8              BNE  PRBL2          LOOP UNTIL COUNT=0
F952:  60                 RTS
F953:  38        PCADJ    SEC                 0=1-BYTE,1=2-BYTE,
F954:  A5 2F     PCADJ2   LDA  LENGTH           2=3-BYTE
F956:  A4 3B     PCADJ3   LDY  PCH
F958:  AA                 TAX                 TEST DISPLACEMENT SIGN
F959:  10 01              BPL  PCADJ4           (FOR REL BRANCH)
F95B:  88                 DEY                 EXTEND NEG BY DECR PCH
F95C:  65 3A     PCADJ4   ADC  PCL
F95E:  90 01              BCC  RTS2           PCL+LENGTH(OR DISPL)+1 TO A
F960:  C8                 INY                   CARRY INTO Y (PCH)
F961:  60        RTS2     RTS
                 *        FMT1 BYTES:           XXXXXXY0 INSTRS
                 *        IF Y=0                THEN LEFT HALF BYTE
                 *        IF Y=1                THEN RIGHT HALF BYTE
                 *                                   (X=INDEX)
F962:  04 20 54
F965:  30 0D     FMT1     DFB  $04,$20,$54,$
F967:  80 04 90
F96A:  03 22              DFB  $80,$04,$90,$
F96C:  54 33 0D
F96F:  80 04              DFB  $54,$33,$0D,$
F971:  90 04 20
F974:  54 33              DFB  $90,$04,$20,$
F976:  0D 80 04
F979:  90 04              DFB  $0D,$80,$04,$
F97B:  20 54 3B
F97E:  0D 80              DFB  $20,$54,$3B,$
F980:  04 90 00
F983:  22 44              DFB  $04,$90,$00,$
F985:  33 0D C8
F988:  44 00              DFB  $33,$0D,$C8,$
F98A:  11 22 44
F98D:  33 0D              DFB  $11,$22,$44,$
F98F:  C8 44 A9
F992:  01 22              DFB  $C8,$44,$A9,$
F994:  44 33 0D
F997:  80 04              DFB  $44,$33,$0D,$
F999:  90 01 22
F99C:  44 33              DFB  $90,$01,$22,$
F99E:  0D 80 04
F9A1:  90                 DFB  $0D,$80,$04,$
F9A2:  26 31 87
F9A5:  9A                 DFB  $26,$31,$87,$ZZXXXY01 INSTR'S
F9A6:  00        FMT2     DFB  $00            ERR
F9A7:  21                 DFB  $21            IMM
F9A8:  81                 DFB  $81            Z-PAGE
F9A9:  82                 DFB  $82            ABS
F9AA:  00                 DFB  $00            IMPLIED
F9AB:  00                 DFB  $00            ACCUMULATOR
F9AC:  59                 DFB  $59            (ZPAG,X)
F9AD:  4D                 DFB  $4D            (ZPAG),Y
F9AE:  91                 DFB  $91            ZPAG,X
F9AF:  92                 DFB  $92            ABS,X
F9B0:  86                 DFB  $86            ABS,Y
F9B1:  4A                 DFB  $4A            (ABS)
F9B2:  85                 DFB  $85            ZPAG,Y
F9B3:  9D                 DFB  $9D            RELATIVE
F9B4:  AC A9 AC
F9B7:  A3 A8 A4  CHAR1    ASC  ",),#($"
F9BA:  D9 00 D8
F9BD:  A4 A4 00  CHAR2    DFB  $D9,$00,$D8,$
                 *CHAR2:  "Y",0,"X$$",0
                 *        MNEML          IS OF FORM:
                 *        (A)  XXXXX000
                 *        (B)  XXXYY100
                 *        (C)  1XXX1010
                 *        (D)  XXXYYY10
                 *        (E)  XXXYYY01
                 *             (X=INDEX)
F9C0:  1C 8A 1C
F9C3:  23 5D 8B  MNEML    DFB  $1C,$8A,$1C,$
F9C6:  1B A1 9D
```

```
F9C9:  8A 1D 23                DFB   $1B,$A1,$9D,$
F9CC:  9D 8B 1D
F9CF:  A1 00 29                DFB   $9D,$8B,$1D,$
F9D2:  19 AE 69
F9D5:  A8 19 23                DFB   $19,$AE,$69,$
F9D8:  24 53 1B
F9DB:  23 24 53                DFB   $24,$53,$1B,$
F9DE:  19 A1                   DFB   $19,$A1         (A) FORMAT ABOVE
F9E0:  00 1A 5B
F9E3:  5B A5 69                DFB   $00,$1A,$5B,$
F9E6:  24 24                   DFB   $24,$24         (B) FORMAT
F9E8:  AE AE A8
F9EB:  AD 29 00                DFB   $AE,$AE,$A8,$
F9EE:  7C 00                   DFB   $7C,$00         (C) FORMAT
F9F0:  15 9C 6D
F9F3:  9C A5 69                DFB   $15,$9C,$6D,$
F9F6:  29 53                   DFB   $29,$53         (D) FORMAT
F9F8:  84 13 34
F9FB:  11 A5 69                DFB   $84,$13,$34,$
F9FE:  23 A0                   DFB   $23,$A0         (E) FORMAT
FA00:  D8 62 5A
FA03:  48 26 62     MNEMR      DFB   $D8,$62,$5A,$
FA06:  94 88 54
FA09:  44 C8 54                DFB   $94,$88,$54,$
FA0C:  68 44 E8
FA0F:  94 00 B4                DFB   $68,$44,$E8,$
FA12:  08 84 74
FA15:  B4 28 6E                DFB   $08,$84,$74,$
FA18:  74 F4 CC
FA1B:  4A 72 F2                DFB   $74,$F4,$CC,$
FA1E:  A4 8A                   DFB   $A4,$8A         (A) FORMAT
FA20:  00 AA A2
FA23:  A2 74 74                DFB   $00,$AA,$A2,$
FA26:  74 72                   DFB   $74,$72         (B) FORMAT
FA28:  44 68 B2
FA2B:  32 B2 00                DFB   $44,$68,$B2,$
FA2E:  22 00                   DFB   $22,$00         (C) FORMAT
FA30:  1A 1A 26
FA33:  26 72 72                DFB   $1A,$1A,$26,$
FA36:  88 C8                   DFB   $88,$C8         (D) FORMAT
FA38:  C4 CA 26
FA3B:  48 44 44                DFB   $C4,$CA,$26,$
FA3E:  A2 C8                   DFB   $A2,$C8         (E) FORMAT
FA40:  FF FF FF                DFB   $FF,$FF,$FF
FA43:  20 D0 F8     STEP       JSR   INSTDSP         DISASSEMBLE ONE INST
FA46:  68                      PLA                     AT (PCL,H)
FA47:  85 2C                   STA   RTNL            ADJUST TO USER
FA49:  68                      PLA                     STACK. SAVE
FA4A:  85 2D                   STA   RTNH              RTN ADR.
FA4C:  A2 08                   LDX   #$08
FA4E:  BD 10 FB     XQINIT     LDA   INITBL-1,X      INIT XEQ AREA
FA51:  95 3C                   STA   XQT,X
FA53:  CA                      DEX
FA54:  D0 F8                   BNE   XQINIT
FA56:  A1 3A                   LDA   (PCL,X)         USER OPCODE BYTE
FA58:  F0 42                   BEQ   XBRK            SPECIAL IF BREAK
FA5A:  A4 2F                   LDY   LENGTH          LEN FROM DISASSEMBLY
FA5C:  C9 20                   CMP   #$20
FA5E:  F0 59                   BEQ   XJSR            HANDLE JSR, RTS, JMP,
FA60:  C9 60                   CMP   #$60              JMP ( ), RTI SPECIAL
FA62:  F0 45                   BEQ   XRTS
FA64:  C9 4C                   CMP   #$4C
FA66:  F0 5C                   BEQ   XJMP
FA68:  C9 6C                   CMP   #$6C
FA6A:  F0 59                   BEQ   XJMPAT
FA6C:  C9 40                   CMP   #$40
FA6E:  F0 35                   BEQ   XRTI
FA70:  29 1F                   AND   #$1F
FA72:  49 14                   EOR   #$14
FA74:  C9 04                   CMP   #$04            COPY USER INST TO XEQ AREA
FA76:  F0 02                   BEQ   XQ2               WITH TRAILING NOPS
FA78:  B1 3A        XQ1        LDA   (PCL),Y         CHANGE REL BRANCH
FA7A:  99 3C 00     XQ2        STA   XQTNZ,Y           DISP TO 4 FOR
```

```
FA7D:  88               DEY               JMP TO BRANCH OR
FA7E:  10 F8            BPL    XQ1        NBRANCH FROM XEQ.
FA80:  20 3F FF         JSR    RESTORE   RESTORE USER REG CONTENTS
FA83:  4C 3C 00         JMP    XQTNZ     XEQ USER OP FROM RAM
FA86:  85 45     IRQ    STA    ACC          (RETURN TO NBRANCH)
FA88:  68               PLA
FA89:  48               PHA              **IRQ HANDLER
FA8A:  0A               ASL    A
FA8B:  0A               ASL    A
FA8C:  0A               ASL    A
FA8D:  30 03            BMI    BREAK     TEST FOR BREAK
FA8F:  6C FE 03         JMP    (IRQLOC)  USER ROUTINE VECTOR IN RAM
FA92:  28        BREAK  PLP
FA93:  20 4C FF         JSR    SAV1      SAVE REG'S ON BREAK
FA96:  68               PLA                INCLUDING PC
FA97:  85 3A            STA    PCL
FA99:  68               PLA
FA9A:  85 3B            STA    PCH
FA9C:  20 82 F8  XBRK   JSR    INSDS1    PRINT USER PC.
FA9F:  20 DA FA         JSR    RGDSP1      AND REG'S
FAA2:  4C 65 FF         JMP    MON       GO TO MONITOR
FAA5:  18        XRTI   CLC
FAA6:  68               PLA              SIMULATE RTI BY EXPECTING
FAA7:  85 48            STA    STATUS      STATUS FROM STACK, THEN RTS
FAA9:  68        XRTS   PLA              RTS SIMULATION
FAAA:  85 3A            STA    PCL         EXTRACT PC FROM STACK
FAAC:  68               PLA                AND UPDATE PC BY 1 (LEN=0)
FAAD:  85 3B     PCINC2 STA    PCH
FAAF:  A5 2F     PCINC3 LDA    LENGTH    UPDATE PC BY LEN
FAB1:  20 56 F9         JSR    PCADJ3
FAB4:  84 3B            STY    PCH
FAB6:  18               CLC
FAB7:  90 14            BCC    NEWPCL
FAB9:  18        XJSR   CLC
FABA:  20 54 F9         JSR    PCADJ2    UPDATE PC AND PUSH
FABD:  AA               TAX                ONTO STACK FOR
FABE:  98               TYA                JSR SIMULATE
FABF:  48               PHA
FAC0:  8A               TXA
FAC1:  48               PHA
FAC2:  A0 02            LDY    #$02
FAC4:  18        XJMP   CLC
FAC5:  B1 3A     XJMPAT LDA    (PCL),Y
FAC7:  AA               TAX              LOAD PC FOR JMP,
FAC8:  88               DEY                (JMP) SIMULATE.
FAC9:  B1 3A            LDA    (PCL),Y
FACB:  86 3B            STX    PCH
FACD:  85 3A     NEWPCL STA    PCL
FACF:  B0 F3            BCS    XJMP
FAD1:  A5 2D     RTNJMP LDA    RTNH
FAD3:  48               PHA
FAD4:  A5 2C            LDA    RTNL
FAD6:  48               PHA
FAD7:  20 8E FD  REGDSP JSR    CROUT     DISPLAY USER REG
FADA:  A9 45     RGDSP1 LDA    #ACC        CONTENTS WITH
FADC:  85 40            STA    A3L         LABELS
FADE:  A9 00            LDA    #ACC/256
FAE0:  85 41            STA    A3H
FAE2:  A2 FB            LDX    #$FB
FAE4:  A9 A0     RDSP1  LDA    #$A0
FAE6:  20 ED FD         JSR    COUT
FAE9:  BD 1E FA         LDA    RTBL-$FB,X
FAEC:  20 ED FD         JSR    COUT
FAEF:  A9 BD            LDA    #$BD
FAF1:  20 ED FD         JSR    COUT
FAF4:  B5 4A            LDA    ACC+5,X
FAF6:  20 DA FD         JSR    PRBYTE
FAF9:  E8               INX
FAFA:  30 E8            BMI    RDSP1
FAFC:  60               RTS
FAFD:  18        BRANCH CLC              BRANCH TAKEN,
FAFE:  A0 01            LDY    #$01        ADD LEN+2 TO PC
FB00:  B1 3A            LDA    (PCL),Y
```

```
FB02:  20 56 F9            JSR  PCADJ3
FB05:  85 3A               STA  PCL
FB07:  98                  TYA
FB08:  38                  SEC
FB09:  B0 A2               BCS  PCINC2
FB0B:  20 4A FF  NBRNCH    JSR  SAVE         NORMAL RETURN AFTER
FB0E:  38                  SEC                 XEQ USER OF
FB0F:  B0 9E               BCS  PCINC3       GO UPDATE PC
FB11:  EA        INITBL    NOP
FB12:  EA                  NOP               DUMMY FILL FOR
FB13:  4C 0B FB            JMP  NBRNCH         XEQ AREA
FB16:  4C FD FA            JMP  BRANCH
FB19:  C1        RTBL      DFB  $C1
FB1A:  D8                  DFB  $D8
FB1B:  D9                  DFB  $D9
FB1C:  D0                  DFB  $D0
FB1D:  D3                  DFB  $D3
FB1E:  AD 70 C0  PREAD     LDA  PTRIG        TRIGGER PADDLES
FB21:  A0 00               LDY  #$00         INIT COUNT
FB23:  EA                  NOP               COMPENSATE FOR 1ST COUNT
FB24:  EA                  NOP
FB25:  BD 64 C0  PREAD2    LDA  PADDL0,X     COUNT Y-REG EVERY
FB28:  10 04               BPL  RTS2D          12 USEC
FB2A:  C8                  INY
FB2B:  D0 F8               BNE  PREAD2         EXIT AT 255 MAX
FB2D:  88                  DEY
FB2E:  60        RTS2D     RTS
FB2F:  A9 00     INIT      LDA  #$00         CLR STATUS FOR DEBUG
FB31:  85 48               STA  STATUS         SOFTWARE
FB33:  AD 56 C0            LDA  LORES
FB36:  AD 54 C0            LDA  LOWSCR       INIT VIDEO MODE
FB39:  AD 51 C0  SETTXT    LDA  TXTSET       SET FOR TEXT MODE
FB3C:  A9 00               LDA  #$00           FULL SCREEN WINDOW
FB3E:  F0 0B               BEQ  SETWND
FB40:  AD 50 C0  SETGR     LDA  TXTCLR       SET FOR GRAPHICS MODE
FB43:  AD 53 C0            LDA  MIXSET         LOWER 4 LINES AS
FB46:  20 36 F8            JSR  CLRTOP         TEXT WINDOW
FB49:  A9 14               LDA  #$14
FB4B:  85 22     SETWND    STA  WNDTOP       SET FOR 40 COL WINDOW
FB4D:  A9 00               LDA  #$00           TOP IN A-REG,
FB4F:  85 20               STA  WNDLFT         BTTM AT LINE 24
FB51:  A9 28               LDA  #$28
FB53:  85 21               STA  WNDWDTH
FB55:  A9 18               LDA  #$18
FB57:  85 23               STA  WNDBTM         VTAB TO ROW 23
FB59:  A9 17               LDA  #$17
FB5B:  85 25     TABV      STA  CV           VTABS TO ROW IN A-REG
FB5D:  4C 22 FC            JMP  VTAB
FB60:  20 A4 FB  MULPM     JSR  MD1          ABS VAL OF AC AUX
FB63:  A0 10     MUL       LDY  #$10         INDEX FOR 16 BITS
FB65:  A5 50     MUL2      LDA  ACL          ACX * AUX + XTND
FB67:  4A                  LSR  A              TO AC, XTND
FB68:  90 0C               BCC  MUL4         IF NO CARRY,
FB6A:  18                  CLC                 NO PARTIAL PROD.
FB6B:  A2 FE               LDX  #$FE
FB6D:  B5 54     MUL3      LDA  XTNDL+2,X    ADD MPLCND (AUX)
FB6F:  75 56               ADC  AUXL+2,X      TO PARTIAL PROD
FB71:  95 54               STA  XTNDL+2,X         (XTND).
FB73:  E8                  INX
FB74:  D0 F7               BNE  MUL3
FB76:  A2 03     MUL4      LDX  #$03
FB78:  76        MUL5      DFB  #$76
FB79:  50                  DFB  #$50
FB7A:  CA                  DEX
FB7B:  10 FB               BPL  MUL5
FB7D:  88                  DEY
FB7E:  D0 E5               BNE  MUL2
FB80:  60                  RTS
FB81:  20 A4 FB  DIVPM     JSR  MD1          ABS VAL OF AC, AUX.
FB84:  A0 10     DIV       LDY  #$10         INDEX FOR 16 BITS
FB86:  06 50     DIV2      ASL  ACL
FB88:  26 51               ROL  ACH
FB8A:  26 52               ROL  XTNDL        XTND/AUX
```

```
FB8C:  26 53              ROL  XTNDH          TO AC.
FB8E:  38                 SEC
FB8F:  A5 52              LDA  XTNDL
FB91:  E5 54              SBC  AUXL         MOD TO XTND.
FB93:  AA                 TAX
FB94:  A5 53              LDA  XTNDH
FB96:  E5 55              SBC  AUXH
FB98:  90 06              BCC  DIV3
FB9A:  86 52              STX  XTNDL
FB9C:  85 53              STA  XTNDH
FB9E:  E6 50              INC  ACL
FBA0:  88        DIV3     DEY
FBA1:  D0 E3              BNE  DIV2
FBA3:  60                 RTS
FBA4:  A0 00     MD1      LDY  #$00         ABS VAL OF AC, AUX
FBA6:  84 2F              STY  SIGN           WITH RESULT SIGN
FBA8:  A2 54              LDX  #AUXL          IN LSB OF SIGN.
FBAA:  20 AF FB           JSR  MD2
FBAD:  A2 50              LDX  #ACL
FBAF:  B5 01     MD2      LDA  LOC1,X       X SPECIFIES AC OR AUX
FBB1:  10 0D              BPL  MDRTS
FBB3:  38                 SEC
FBB4:  98        MD3      TYA
FBB5:  F5 00              SBC  LOC0,X       COMPL SPECIFIED REG
FBB7:  95 00              STA  LOC0,X         IF NEG.
FBB9:  98                 TYA
FBBA:  F5 01              SBC  LOC1,X
FBBC:  95 01              STA  LOC1,X
FBBE:  E6 2F              INC  SIGN
FBC0:  60        MDRTS    RTS
FBC1:  48        BASCALC  PHA               CALC BASE ADR IN BASL,H
FBC2:  4A                 LSR  A              FOR GIVEN LINE NO.
FBC3:  29 03              AND  #$03           0<=LINE NO.<=$17
FBC5:  09 04              ORA  #$04         ARG=000ABCDE, GENERATE
FBC7:  85 29              STA  BASH           BASH=000001CD
FBC9:  68                 PLA                 AND
FBCA:  29 18              AND  #$18           BASL=EABAB000
FBCC:  90 02              BCC  BSCLC2
FBCE:  69 7F              ADC  #$7F
FBD0:  85 28     BSCLC2   STA  BASL
FBD2:  0A                 ASL  A
FBD3:  0A                 ASL  A
FBD4:  05 28              ORA  BASL
FBD6:  85 28              STA  BASL
FBD8:  60                 RTS
FBD9:  C9 87     BELL1    CMP  #$87         BELL CHAR? (CNTRL-G)
FBDB:  D0 12              BNE  RTS2B          NO,RETURN
FBDD:  A9 40              LDA  #$40         DELAY .01 SECONDS
FBDF:  20 A8 FC           JSR  WAIT
FBE2:  A0 C0              LDY  #$C0
FBE4:  A9 0C     BELL2    LDA  #$0C         TOGGLE SPEAKER AT
FBE6:  20 A8 FC           JSR  WAIT           1 KHZ FOR .1 SEC.
FBE9:  AD 30 C0           LDA  SPKR
FBEC:  88                 DEY
FBED:  D0 F5              BNE  BELL2
FBEF:  60        RTS2B    RTS
FBF0:  A4 24     STOADV   LDY  CH           CURSER H INDEX TO Y-REG
FBF2:  91 28              STA  (BASL),Y     STOR CHAR IN LINE
FBF4:  E6 24     ADVANCE  INC  CH           INCREMENT CURSER H INDEX
FBF6:  A5 24              LDA  CH             (MOVE RIGHT)
FBF8:  C5 21              CMP  WNDWDTH      BEYOND WINDOW WIDTH?
FBFA:  B0 66              BCS  CR             YES CR TO NEXT LINE
FBFC:  60        RTS3     RTS                 NO,RETURN
FBFD:  C9 A0     VIDOUT   CMP  #$A0         CONTROL CHAR?
FBFF:  B0 EF              BCS  STOADV         NO,OUTPUT IT.
FC01:  A8                 TAY               INVERSE VIDEO?
FC02:  10 EC              BPL  STOADV         YES, OUTPUT IT.
FC04:  C9 8D              CMP  #$8D         CR?
FC06:  F0 5A              BEQ  CR             YES.
FC08:  C9 8A              CMP  #$8A         LINE FEED?
FC0A:  F0 5A              BEQ  LF             IF SO, DO IT.
FC0C:  C9 88              CMP  #$88         BACK SPACE? (CNTRL-H)
FC0E:  D0 C9              BNE  BELL1          NO, CHECK FOR BELL.
```

```
FC10:  C6 24      BS       DEC  CH          DECREMENT CURSER H INDEX
FC12:  10 E8               BPL  RTS3        IF POS, OK. ELSE MOVE UP
FC14:  A5 21               LDA  WNDWDTH     SET CH TO WNDWDTH-1
FC16:  85 24               STA  CH
FC18:  C6 24               DEC  CH          (RIGHTMOST SCREEN POS)
FC1A:  A5 22      UP       LDA  WNDTOP      CURSER V INDEX
FC1C:  C5 25               CMP  CV
FC1E:  B0 0B               BCS  RTS4        IF TOP LINE THEN RETURN
FC20:  C6 25               DEC  CV          DECR CURSER V-INDEX
FC22:  A5 25      VTAB     LDA  CV          GET CURSER V-INDEX
FC24:  20 C1 FB   VTABZ    JSR  BASCALC     GENERATE BASE ADDR
FC27:  65 20               ADC  WNDLFT      ADD WINDOW LEFT INDEX
FC29:  85 28               STA  BASL        TO BASL
FC2B:  60         RTS4     RTS
FC2C:  49 C0      ESC1     EOR  #$C0        ESC?
FC2E:  F0 28               BEQ  HOME          IF SO, DO HOME AND CLEAR
FC30:  69 FD               ADC  #$FD        ESC-A OR B CHECK
FC32:  90 C0               BCC  ADVANCE       A, ADVANCE
FC34:  F0 DA               BEQ  BS            B, BACKSPACE
FC36:  69 FD               ADC  #$FD        ESC-C OR D CHECK
FC38:  90 2C               BCC  LF            C,DOWN
FC3A:  F0 DE               BEQ  UP            D, GO UP
FC3C:  69 FD               ADC  #$FD        ESC-E OR F CHECK
FC3E:  90 5C               BCC  CLREOL        E, CLEAR TO END OF LINE
FC40:  D0 E9               BNE  RTS4          NOT F, RETURN
FC42:  A4 24      CLREOP   LDY  CH          CURSOR H TO Y INDEX
FC44:  A5 25               LDA  CV          CURSOR V TO A-REGISTER
FC46:  48         CLEOP1   PHA              SAVE CURRENT LINE ON STK
FC47:  20 24 FC            JSR  VTABZ       CALC BASE ADDRESS
FC4A:  20 9E FC            JSR  CLEOLZ      CLEAR TO EOL, SET CARRY
FC4D:  A0 00               LDY  #$00        CLEAR FROM H INDEX=0 FOR REST
FC4F:  68                  PLA              INCREMENT CURRENT LINE
FC50:  69 00               ADC  #$00        (CARRY IS SET)
FC52:  C5 23               CMP  WNDBTM      DONE TO BOTTOM OF WINDOW?
FC54:  90 F0               BCC  CLEOP1        NO, KEEP CLEARING LINES
FC56:  B0 CA               BCS  VTAB          YES, TAB TO CURRENT LINE
FC58:  A5 22      HOME     LDA  WNDTOP      INIT CURSOR V
FC5A:  85 25               STA  CV            AND H-INDICES
FC5C:  A0 00               LDY  #$00
FC5E:  84 24               STY  CH          THEN CLEAR TO END OF PAGE
FC60:  F0 E4               BEQ  CLEOP1
FC62:  A9 00      CR       LDA  #$00        CURSOR TO LEFT OF INDEX
FC64:  85 24               STA  CH          (RET CURSOR H=0)
FC66:  E6 25      LF       INC  CV          INCR CURSOR V(DOWN 1 LINE)
FC68:  A5 25               LDA  CV
FC6A:  C5 23               CMP  WNDBTM      OFF SCREEN?
FC6C:  90 B6               BCC  VTABZ         NO, SET BASE ADDR
FC6E:  C6 25               DEC  CV          DECR CURSOR V(BACK TO BOTTOM)
FC70:  A5 22      SCROLL   LDA  WNDTOP      START AT TOP OF SCRL WNDW
FC72:  48                  PHA
FC73:  20 24 FC            JSR  VTABZ       GENERATE BASE ADDRESS
FC76:  A5 28      SCRL1    LDA  BASL        COPY BASL,H
FC78:  85 2A               STA  BAS2L         TO BAS2L,H
FC7A:  A5 29               LDA  BASH
FC7C:  85 2B               STA  BAS2H
FC7E:  A4 21               LDY  WNDWDTH     INIT Y TO RIGHTMOST INDEX
FC80:  88                  DEY                OF SCROLLING WINDOW
FC81:  68                  PLA
FC82:  69 01               ADC  #$01        INCR LINE NUMBER
FC84:  C5 23               CMP  WNDBTM      DONE?
FC86:  B0 0D               BCS  SCRL3         YES, FINISH
FC88:  48                  PHA
FC89:  20 24 FC            JSR  VTABZ       FORM BASL,H (BASE ADDR)
FC8C:  B1 28      SCRL2    LDA  (BASL),Y    MOVE A CHR UP ON LINE
FC8E:  91 2A               STA  (BAS2L),Y
FC90:  88                  DEY              NEXT CHAR OF LINE
FC91:  10 F9               BPL  SCRL2
FC93:  30 E1               BMI  SCRL1       NEXT LINE
FC95:  A0 00      SCRL3    LDY  #$00        CLEAR BOTTOM LINE
FC97:  20 9E FC            JSR  CLEOLZ      GET BASE ADDR FOR BOTTOM LINE
FC9A:  B0 86               BCS  VTAB        CARRY IS SET
FC9C:  A4 24      CLREOL   LDY  CH          CURSOR H INDEX
FC9E:  A9 A0      CLEOLZ   LDA  #$A0
```

```
FCA0:  91 28      CLEOL2  STA  (BASL),Y    STORE BLANKS FROM 'HERE'
FCA2:  C8                 INY                TO END OF LINES (WNDWDTH)
FCA3:  C4 21              CPY  WNDWDTH
FCA5:  90 F9              BCC  CLEOL2
FCA7:  60                 RTS
FCA8:  38         WAIT    SEC
FCA9:  48         WAIT2   PHA
FCAA:  E9 01      WAIT3   SBC  #$01
FCAC:  D0 FC              BNE  WAIT3       1.0204 USEC
FCAE:  68                 PLA              (13+2712*A+512*A*A)
FCAF:  E9 01              SBC  #$01
FCB1:  D0 F6              BNE  WAIT2
FCB3:  60                 RTS
FCB4:  E6 42      NXTA4   INC  A4L         INCR 2-BYTE A4
FCB6:  D0 02              BNE  NXTA1         AND A1
FCB8:  E6 43              INC  A4H
FCBA:  A5 3C      NXTA1   LDA  A1L         INCR 2-BYTE A1.
FCBC:  C5 3E              CMP  A2L
FCBE:  A5 3D              LDA  A1H           AND COMPARE TO A2
FCC0:  E5 3F              SBC  A2H
FCC2:  E6 3C              INC  A1L           (CARRY SET IF >=)
FCC4:  D0 02              BNE  RTS4B
FCC6:  E6 3D              INC  A1H
FCC8:  60         RTS4B   RTS
FCC9:  A0 4B      HEADR   LDY  #$4B        WRITE A*256 'LONG 1'
FCCB:  20 DB FC           JSR  ZERDLY        HALF CYCLES
FCCE:  D0 F9              BNE  HEADR         (650 USEC EACH )
FCD0:  69 FE              ADC  #$FE
FCD2:  B0 F5              BCS  HEADR       THEN A 'SHORT 0'
FCD4:  A0 21              LDY  #$21          (400 USEC)
FCD6:  20 DB FC   WRBIT   JSR  ZERDLY      WRITE TWO HALF CYCLES
FCD9:  C8                 INY                OF 250 USEC ('0')
FCDA:  C8                 INY                OR 500 USEC ('0')
FCDB:  88         ZERDLY  DEY
FCDC:  D0 FD              BNE  ZERDLY
FCDE:  90 05              BCC  WRTAPE      Y IS COUNT FOR
FCE0:  A0 32              LDY  #$32          TIMING LOOP
FCE2:  88         ONEDLY  DEY
FCE3:  D0 FD              BNE  ONEDLY
FCE5:  AC 20 C0   WRTAPE  LDY  TAPEOUT
FCE8:  A0 2C              LDY  #$2C
FCEA:  CA                 DEX
FCEB:  60                 RTS
FCEC:  A2 08      RDBYTE  LDX  #$08        8 BITS TO READ
FCEE:  48         RDBYT2  PHA              READ TWO TRANSITIONS
FCEF:  20 FA FC           JSR  RD2BIT        (FIND EDGE)
FCF2:  68                 PLA
FCF3:  2A                 ROL  A           NEXT BIT
FCF4:  A0 3A              LDY  #$3A        COUNT FOR SAMPLES
FCF6:  CA                 DEX
FCF7:  D0 F5              BNE  RDBYT2
FCF9:  60                 RTS
FCFA:  20 FD FC   RD2BIT  JSR  RDBIT
FCFD:  88         RDBIT   DEY              DECR Y UNTIL
FCFE:  AD 60 C0           LDA  TAPEIN        TAPE TRANSITION
FD01:  45 2F              EOR  LASTIN
FD03:  10 F8              BPL  RDBIT
FD05:  45 2F              EOR  LASTIN
FD07:  85 2F              STA  LASTIN
FD09:  C0 80              CPY  #$80        SET CARRY ON Y-REG.
FD0B:  60                 RTS
FD0C:  A4 24      RDKEY   LDY  CH
FD0E:  B1 28              LDA  (BASL),Y    SET SCREEN TO FLASH
FD10:  48                 PHA
FD11:  29 3F              AND  #$3F
FD13:  09 40              ORA  #$40
FD15:  91 28              STA  (BASL),Y
FD17:  68                 PLA
FD18:  6C 38 00           JMP  (KSWL)      GO TO USER KEY-IN
FD1B:  E6 4E      KEYIN   INC  RNDL
FD1D:  D0 02              BNE  KEYIN2      INCR RND NUMBER
FD1F:  E6 4F              INC  RNDH
FD21:  2C 00 C0   KEYIN2  BIT  KBD         KEY DOWN?
```

```
FD24:  10 F5        BPL  KEYIN      LOOP
FD26:  91 28        STA  (BASL),Y   REPLACE FLASHING SCREEN
FD28:  AD 00 C0     LDA  KBD        GET KEYCODE
FD2B:  2C 10 C0     BIT  KBDSTRB    CLR KEY STROBE
FD2E:  60           RTS
FD2F:  20 0C FD  ESC       JSR  RDKEY      GET KEYCODE
FD32:  20 2C FC           JSR  ESC1         HANDLE ESC FUNC.
FD35:  20 0C FD  RDCHAR    JSR  RDKEY      READ KEY
FD38:  C9 9B              CMP  #$9B       ESC?
FD3A:  F0 F3              BEQ  ESC          YES, DON'T RETURN
FD3C:  60                 RTS
FD3D:  A5 32     NOTCR     LDA  INVFLG
FD3F:  48                 PHA
FD40:  A9 FF              LDA  #$FF
FD42:  85 32              STA  INVFLG     ECHO USER LINE
FD44:  BD 00 02           LDA  IN,X         NON INVERSE
FD47:  20 ED FD           JSR  COUT
FD4A:  68                 PLA
FD4B:  85 32              STA  INVFLG
FD4D:  BD 00 02           LDA  IN,X
FD50:  C9 88              CMP  #$88       CHECK FOR EDIT KEYS
FD52:  F0 1D              BEQ  BCKSPC       BS, CTRL-X.
FD54:  C9 98              CMP  #$98
FD56:  F0 0A              BEQ  CANCEL
FD58:  E0 F8              CPX  #$F8       MARGIN?
FD5A:  90 03              BCC  NOTCR1
FD5C:  20 3A FF           JSR  BELL         YES, SOUND BELL
FD5F:  E8        NOTCR1    INX              ADVANCE INPUT INDEX
FD60:  D0 13              BNE  NXTCHAR
FD62:  A9 DC     CANCEL    LDA  #$DC       BACKSLASH AFTER CANCELLED LIN
FD64:  20 ED FD           JSR  COUT
FD67:  20 8E FD  GETLNZ    JSR  CROUT      OUTPUT CR
FD6A:  A5 33     GETLN     LDA  PROMPT
FD6C:  20 ED FD           JSR  COUT       OUTPUT PROMPT CHAR
FD6F:  A2 01              LDX  #$01       INIT INPUT INDEX
FD71:  8A        BCKSPC    TXA                WILL BACKSPACE TO 0
FD72:  F0 F3              BEQ  GETLNZ
FD74:  CA                 DEX
FD75:  20 35 FD  NXTCHAR   JSR  RDCHAR
FD78:  C9 95              CMP  #PICK      USE SCREEN CHAR
FD7A:  D0 02              BNE  CAPTST       FOR CTRL-U
FD7C:  B1 28              LDA  (BASL),Y
FD7E:  C9 E0     CAPTST    CMP  #$E0
FD80:  90 02              BCC  ADDINP     CONVERT TO CAPS
FD82:  29 DF              AND  #$DF
FD84:  9D 00 02  ADDINP    STA  IN,X       ADD TO INPUT BUF
FD87:  C9 8D              CMP  #$8D
FD89:  D0 B2              BNE  NOTCR
FD8B:  20 9C FC           JSR  CLREOL     CLR TO EOL IF CR
FD8E:  A9 8D     CROUT     LDA  #$8D
FD90:  D0 5B              BNE  COUT
FD92:  A4 3D     PRA1      LDY  A1H        PRINT CR,A1 IN HEX
FD94:  A6 3C              LDX  A1L
FD96:  20 8E FD  PRYX2     JSR  CROUT
FD99:  20 40 F9           JSR  PRNTYX
FD9C:  A0 00              LDY  #$00
FD9E:  A9 AD              LDA  #$AD       PRINT '-'
FDA0:  4C ED FD           JMP  COUT
FDA3:  A5 3C     XAM8      LDA  A1L
FDA5:  09 07              ORA  #$07       SET TO FINISH AT
FDA7:  85 3E              STA  A2L          MOD 8=7
FDA9:  A5 3D              LDA  A1H
FDAB:  85 3F              STA  A2H
FDAD:  A5 3C     MOD8CHK   LDA  A1L
FDAF:  29 07              AND  #$07
FDB1:  D0 03              BNE  DATAOUT
FDB3:  20 92 FD  XAM       JSR  PRA1
FDB6:  A9 A0     DATAOUT   LDA  #$A0
FDB8:  20 ED FD           JSR  COUT       OUTPUT BLANK
FDBB:  B1 3C              LDA  (A1L),Y
FDBD:  20 DA FD           JSR  PRBYTE     OUTPUT BYTE IN HEX
FDC0:  20 BA FC           JSR  NXTA1
```

```
FDC3:   90 E6              BCC   MOD8CHK      CHECK IF TIME TO,
FDC5:   60        RTS4C    RTS                  PRINT ADDR
FDC6:   4A        XAMPM    LSR   A            DETERMINE IF MON
FDC7:   90 EA              BCC   XAM            MODE IS XAM
FDC9:   4A                 LSR   A              ADD, OR SUB
FDCA:   4A                 LSR   A
FDCB:   A5 3E              LDA   A2L
FDCD:   90 02              BCC   ADD
FDCF:   49 FF              EOR   #$FF         SUB: FORM 2'S COMPLEMENT
FDD1:   65 3C     ADD      ADC   A1L
FDD3:   48                 PHA
FDD4:   A9 BD              LDA   #$BD
FDD6:   20 ED FD           JSR   COUT         PRINT '=', THEN RESULT
FDD9:   68                 PLA
FDDA:   48        PRBYTE   PHA                PRINT BYTE AS 2 HEX
FDDB:   4A                 LSR   A              DIGITS, DESTROYS A-REG
FDDC:   4A                 LSR   A
FDDD:   4A                 LSR   A
FDDE:   4A                 LSR   A
FDDF:   20 E5 FD           JSR   PRHEXZ
FDE2:   68                 PLA
FDE3:   29 0F     PRHEX    AND   #$0F         PRINT HEX DIG IN A-REG
FDE5:   09 B0     PRHEXZ   ORA   #$B0           LSB'S
FDE7:   C9 BA              CMP   #$BA
FDE9:   90 02              BCC   COUT
FDEB:   69 06              ADC   #$06
FDED:   6C 36 00  COUT     JMP   (CSWL)       VECTOR TO USER OUTPUT ROUTINE
FDF0:   C9 A0     COUT1    CMP   #$A0
FDF2:   90 02              BCC   COUTZ        DON'T OUTPUT CTRL'S INVERSE
FDF4:   25 32              AND   INVFLG       MASK WITH INVERSE FLAG
FDF6:   84 35     COUTZ    STY   YSAV1        SAV Y-REG
FDF8:   48                 PHA                SAV A-REG
FDF9:   20 FD FB           JSR   VIDOUT       OUTPUT A-REG AS ASCII
FDFC:   68                 PLA                RESTORE A-REG
FDFD:   A4 35              LDY   YSAV1          AND Y-REG
FDFF:   60                 RTS                  THEN RETURN
FE00:   C6 34     BL1      DEC   YSAV
FE02:   F0 9F              BEQ   XAM8
FE04:   CA        BLANK    DEX                BLANK TO MON
FE05:   D0 16              BNE   SETMDZ       AFTER BLANK
FE07:   C9 BA              CMP   #$BA         DATA STORE MODE?
FE09:   D0 BB              BNE   XAMPM          NO, XAM, ADD OR SUB
FE0B:   85 31     STOR     STA   MODE         KEEP IN STORE MODE
FE0D:   A5 3E              LDA   A2L
FE0F:   91 40              STA   (A3L),Y      STORE AS LOW BYTE AS (A3)
FE11:   E6 40              INC   A3L
FE13:   D0 02              BNE   RTS5         INCR A3, RETURN
FE15:   E6 41              INC   A3H
FE17:   60        RTS5     RTS
FE18:   A4 34     SETMODE  LDY   YSAV         SAVE CONVERTED ':', '+',
FE1A:   B9 FF 01           LDA   IN-1,Y         '-', '.' AS MODE.
FE1D:   85 31     SETMDZ   STA   MODE
FE1F:   60                 RTS
FE20:   A2 01     LT       LDX   #$01
FE22:   B5 3E     LT2      LDA   A2L,X        COPY A2 (2 BYTES) TO
FE24:   95 42              STA   A4L,X          A4 AND A5
FE26:   95 44              STA   A5L,X
FE28:   CA                 DEX
FE29:   10 F7              BPL   LT2
FE2B:   60                 RTS
FE2C:   B1 3C     MOVE     LDA   (A1L),Y      MOVE (A1 TO A2) TO
FE2E:   91 42              STA   (A4L),Y        (A4)
FE30:   20 B4 FC           JSR   NXTA4
FE33:   90 F7              BCC   MOVE
FE35:   60                 RTS
FE36:   B1 3C     VFY      LDA   (A1L),Y      VERIFY (A1 TO A2) WITH
FE38:   D1 42              CMP   (A4L),Y        (A4)
FE3A:   F0 1C              BEQ   VFYOK
FE3C:   20 92 FD           JSR   PRA1
FE3F:   B1 3C              LDA   (A1L),Y
FE41:   20 DA FD           JSR   PRBYTE
FE44:   A9 A0              LDA   #$A0
FE46:   20 ED FD           JSR   COUT
```

```
FE49:  A9 A8           LDA  #$A8
FE4B:  20 ED FD        JSR  COUT
FE4E:  B1 42           LDA  (A4L),Y
FE50:  20 DA FD        JSR  PRBYTE
FE53:  A9 A9           LDA  #$A9
FE55:  20 ED FD        JSR  COUT
FE58:  20 B4 FC  VFYOK JSR  NXTA4
FE5B:  90 D9           BCC  VFY
FE5D:  60              RTS
FE5E:  20 75 FE  LIST  JSR  A1PC       MOVE A1 (2 BYTES) TO
FE61:  A9 14           LDA  #$14         PC IF SPEC'D AND
FE63:  48        LIST2 PHA               DISSEMBLE 20 INSTRS
FE64:  20 D0 F8        JSR  INSTDSP
FE67:  20 53 F9        JSR  PCADJ      ADJUST PC EACH INSTR
FE6A:  85 3A           STA  PCL
FE6C:  84 3B           STY  PCH
FE6E:  68              PLA
FE6F:  38              SEC
FE70:  E9 01           SBC  #$01       NEXT OF 20 INSTRS
FE72:  D0 EF           BNE  LIST2
FE74:  60              RTS
FE75:  8A        A1PC  TXA             IF USER SPEC'D ADR
FE76:  F0 07           BEQ  A1PCRTS      COPY FROM A1 TO PC
FE78:  B5 3C     A1PCLP LDA A1L,X
FE7A:  95 3A           STA  PCL,X
FE7C:  CA              DEX
FE7D:  10 F9           BPL  A1PCLP
FE7F:  60        A1PCRTS RTS
FE80:  A0 3F     SETINV LDY #$3F       SET FOR INVERSE VID
FE82:  D0 02           BNE  SETIFLG      VIA COUT1
FE84:  A0 FF     SETNORM LDY #$FF      SET FOR NORMAL VID
FE86:  84 32     SETIFLG STY INVFLG
FE88:  60              RTS
FE89:  A9 00     SETKBD LDA #$00       SIMULATE PORT #0 INPUT
FE8B:  85 3E     INPORT STA A2L          SPECIFIED (KEYIN ROUTINE
FE8D:  A2 38     INPRT LDX  #KSWL
FE8F:  A0 1B           LDY  #KEYIN
FE91:  D0 08           BNE  IOPRT
FE93:  A9 00     SETVID LDA #$00       SIMULATE PORT #0 OUTPUT
FE95:  85 3E     OUTPORT STA A2L         SPECIFIED (COUT1 ROUTINE
FE97:  A2 36     OUTPRT LDX #CSWL
FE99:  A0 F0           LDY  #COUT1
FE9B:  A5 3E     IOPRT LDA  A2L        SET RAM IN/OUT VECTORS
FE9D:  29 0F           AND  #$0F
FE9F:  F0 06           BEQ  IOPRT1
FEA1:  09 C0           ORA  #IOADR/256
FEA3:  A0 00           LDY  #$00
FEA5:  F0 02           BEQ  IOPRT2
FEA7:  A9 FD     IOPRT1 LDA #COUT1/256
FEA9:  94 00     IOPRT2 STY LOC0,X
FEAB:  95 01           STA  LOC1,X
FEAD:  60              RTS
FEAE:  EA              NOP
FEAF:  EA              NOP
FEB0:  4C 00 E0  XBASIC JMP BASIC      TO BASIC WITH SCRATCH
FEB3:  4C 03 E0  BASCONT JMP BASIC2    CONTINUE BASIC
FEB6:  20 75 FE  GO    JSR  A1PC       ADR TO PC IF SPEC'D
FEB9:  20 3F FF        JSR  RESTORE    RESTORE META REGS
FEBC:  6C 3A 00        JMP  (PCL)      GO TO USER SUBR
FEBF:  4C D7 FA  REGZ  JMP  REGDSP     TO REG DISPLAY
FEC2:  C6 34     TRACE DEC  YSAV
FEC4:  20 75 FE  STEPZ JSR  A1PC       ADR TO PC IF SPEC'D
FEC7:  4C 43 FA        JMP  STEP       TAKE ONE STEP
FECA:  4C F8 03  USR   JMP  USRADR     TO USR SUBR AT USRADR
FECD:  A9 40     WRITE LDA  #$40
FECF:  20 C9 FC        JSR  HEADR      WRITE 10-SEC HEADER
FED2:  A0 27           LDY  #$27
FED4:  A2 00     WR1   LDX  #$00
FED6:  41 3C           EOR  (A1L,X)
FED8:  48              PHA
FED9:  A1 3C           LDA  (A1L,X)
```

```
FEDB:  20 ED FE          JSR  WRBYTE
FEDE:  20 BA FC          JSR  NXTA1
FEE1:  A0 1D             LDY  #$1D
FEE3:  68                PLA
FEE4:  90 EE             BCC  WR1
FEE6:  A0 22             LDY  #$22
FEE8:  20 ED FE          JSR  WRBYTE
FEEB:  F0 4D             BEQ  BELL
FEED:  A2 10    WRBYTE   LDX  #$10
FEEF:  0A       WRBYT2   ASL  A
FEF0:  20 D6 FC          JSR  WRBIT
FEF3:  D0 FA             BNE  WRBYT2
FEF5:  60                RTS
FEF6:  20 00 FE CRMON    JSR  BL1          HANDLE CR AS BLANK
FEF9:  68                PLA                 THEN POP STACK
FEFA:  68                PLA                 AND RTN TO MON
FEFB:  D0 6C             BNE  MONZ
FEFD:  20 FA FC READ     JSR  RD2BIT       FIND TAPEIN EDGE
FF00:  A9 16             LDA  #$16
FF02:  20 C9 FC          JSR  HEADR        DELAY 3.5 SECONDS
FF05:  85 2E             STA  CHKSUM       INIT CHKSUM=$FF
FF07:  20 FA FC          JSR  RD2BIT       FIND TAPEIN EDGE
FF0A:  A0 24    RD2      LDY  #$24         LOOK FOR SYNC BIT
FF0C:  20 FD FC          JSR  RDBIT          (SHORT 0)
FF0F:  B0 F9             BCS  RD2            LOOP UNTIL FOUND
FF11:  20 FD FC          JSR  RDBIT        SKIP SECOND SYNC H-CYCLE
FF14:  A0 3B             LDY  #$3B         INDEX FOR 0/1 TEST
FF16:  20 EC FC RD3      JSR  RDBYTE       READ A BYTE
FF19:  81 3C             STA  (A1L,X)      STORE AT (A1)
FF1B:  45 2E             EOR  CHKSUM
FF1D:  85 2E             STA  CHKSUM       UPDATE RUNNING CHKSUM
FF1F:  20 BA FC          JSR  NXTA1        INCR A1, COMPARE TO A2
FF22:  A0 35             LDY  #$35         COMPENSATE 0/1 INDEX
FF24:  90 F0             BCC  RD3          LOOP UNTIL DONE
FF26:  20 EC FC          JSR  RDBYTE       READ CHKSUM BYTE
FF29:  C5 2E             CMP  CHKSUM
FF2B:  F0 0D             BEQ  BELL         GOOD, SOUND BELL AND RETURN
FF2D:  A9 C5    PRERR    LDA  #$C5
FF2F:  20 ED FD          JSR  COUT         PRINT "ERR", THEN BELL
FF32:  A9 D2             LDA  #$D2
FF34:  20 ED FD          JSR  COUT
FF37:  20 ED FD          JSR  COUT
FF3A:  A9 87    BELL     LDA  #$87         OUTPUT BELL AND RETURN
FF3C:  4C ED FD          JMP  COUT
FF3F:  A5 48    RESTORE  LDA  STATUS       RESTORE 6502 REG CONTENTS
FF41:  48                PHA                 USED BY DEBUG SOFTWARE
FF42:  A5 45             LDA  ACC
FF44:  A6 46    RESTR1   LDX  XREG
FF46:  A4 47             LDY  YREG
FF48:  28                PLP
FF49:  60                RTS
FF4A:  85 45    SAVE     STA  ACC          SAVE 6502 REG CONTENTS
FF4C:  86 46    SAV1     STX  XREG
FF4E:  84 47             STY  YREG
FF50:  08                PHP
FF51:  68                PLA
FF52:  85 48             STA  STATUS
FF54:  BA                TSX
FF55:  86 49             STX  SPNT
FF57:  D8                CLD
FF58:  60                RTS
FF59:  20 84 FE RESET    JSR  SETNORM      SET SCREEN MODE
FF5C:  20 2F FB          JSR  INIT           AND INIT KBD/SCREEN
FF5F:  20 93 FE          JSR  SETVID         AS I/O DEV'S
FF62:  20 89 FE          JSR  SETKBD
FF65:  D8       MON      CLD               MUST SET HEX MODE!
FF66:  20 3A FF          JSR  BELL
FF69:  A9 AA    MONZ     LDA  #$AA         '*' PROMPT FOR MON
FF6B:  85 33             STA  PROMPT
FF6D:  20 67 FD          JSR  GETLNZ       READ A LINE
FF70:  20 C7 FF          JSR  ZMODE        CLEAR MON MODE, SCAN IDX
FF73:  20 A7 FF NXTITM   JSR  GETNUM       GET ITEM, NON-HEX
FF76:  84 34             STY  YSAV           CHAR IN A-REG
```

```
FF78:  A0 17            LDY  #$17        X-REG=0 IF NO HEX INPUT
FF7A:  88      CHRSRCH  DEY
FF7B:  30 E8            BMI  MON         NOT FOUND, GO TO MON
FF7D:  D9 CC FF         CMP  CHRTBL,Y    FIND CMND CHAR IN TBL
FF80:  D0 F8            BNE  CHRSRCH
FF82:  20 BE FF         JSR  TOSUB       FOUND, CALL CORRESPONDING
FF85:  A4 34            LDY  YSAV          SUBROUTINE
FF87:  4C 73 FF         JMP  NXTITM
FF8A:  A2 03   DIG      LDX  #$03
FF8C:  0A               ASL  A
FF8D:  0A               ASL  A           GOT HEX DIG,
FF8E:  0A               ASL  A             SHIFT INTO A2
FF8F:  0A               ASL  A
FF90:  0A      NXTBIT   ASL  A
FF91:  26 3E            ROL  A2L
FF93:  26 3F            ROL  A2H
FF95:  CA               DEX              LEAVE X=$FF IF DIG
FF96:  10 F8            BPL  NXTBIT
FF98:  A5 31   NXTBAS   LDA  MODE
FF9A:  D0 06            BNE  NXTBS2      IF MODE IS ZERO
FF9C:  B5 3F            LDA  A2H,X         THEN COPY A2 TO
FF9E:  95 3D            STA  A1H,X         A1 AND A3
FFA0:  95 41            STA  A3H,X
FFA2:  E8      NXTBS2   INX
FFA3:  F0 F3            BEQ  NXTBAS
FFA5:  D0 06            BNE  NXTCHR
FFA7:  A2 00   GETNUM   LDX  #$00        CLEAR A2
FFA9:  86 3E            STX  A2L
FFAB:  86 3F            STX  A2H
FFAD:  B9 00 02 NXTCHR  LDA  IN,Y        GET CHAR
FFB0:  C8               INY
FFB1:  49 B0            EOR  #$B0
FFB3:  C9 0A            CMP  #$0A
FFB5:  90 D3            BCC  DIG         IF HEX DIG, THEN
FFB7:  69 88            ADC  #$88
FFB9:  C9 FA            CMP  #$FA
FFBB:  B0 CD            BCS  DIG
FFBD:  60               RTS
FFBE:  A9 FE   TOSUB    LDA  #GO/256     PUSH HIGH-ORDER
FFC0:  48               PHA                SUBR ADR ON STK
FFC1:  B9 E3 FF         LDA  SUBTBL,Y    PUSH LOW ORDER
FFC4:  48               PHA                SUBR ADR ON STK
FFC5:  A5 31            LDA  MODE
FFC7:  A0 00   ZMODE    LDY  #$00        CLR MODE, OLD MODE
FFC9:  84 31            STY  MODE          TO A-REG
FFCB:  60               RTS              GO TO SUBR VIA RTS
FFCC:  BC      CHRTBL   DFB  $BC         F("CTRL-C")
FFCD:  B2               DFB  $B2         F("CTRL-Y")
FFCE:  BE               DFB  $BE         F("CTRL-E")
FFCF:  ED               DFB  $ED         F("T")
FFD0:  EF               DFB  $EF         F("V")
FFD1:  C4               DFB  $C4         F("CTRL-K")
FFD2:  EC               DFB  $EC         F("S")
FFD3:  A9               DFB  $A9         F("CTRL-P")
FFD4:  BB               DFB  $BB         F("CTRL-B")
FFD5:  A6               DFB  $A6         F("-")
FFD6:  A4               DFB  $A4         F("+")
FFD7:  06               DFB  $06         F("M")  (F=EX-OR $B0+$89)
FFD8:  95               DFB  $95         F("<")
FFD9:  07               DFB  $07         F("N")
FFDA:  02               DFB  $02         F("I")
FFDB:  05               DFB  $05         F("L")
FFDC:  F0               DFB  $F0         F("W")
FFDD:  00               DFB  $00         F("G")
FFDE:  EB               DFB  $EB         F("R")
FFDF:  93               DFB  $93         F(":")
FFE0:  A7               DFB  $A7         F(".")
FFE1:  C6               DFB  $C6         F("CR")
FFE2:  99               DFB  $99         F(BLANK)
FFE3:  B2      SUBTBL   DFB  #BASCONT-1
FFE4:  C9               DFB  #USR-1
FFE5:  BE               DFB  #REGZ-1
```

```
FFE6:  C1                  DFB  #TRACE-1
FFE7:  35                  DFB  #VFY-1
FFE8:  6C                  DFB  #INPRT-1
FFE9:  C3                  DFB  #STEPZ-1
FFEA:  96                  DFB  #OUTPRT-1
FFEB:  AF                  DFB  #XBASIC-1
FFEC:  17                  DFB  #SETMODE-1
FFED:  17                  DFB  #SETMODE-1
FFEE:  2B                  DFB  #MOVE-1
FFEF:  1F                  DFB  #LT-1
FFF0:  83                  DFB  #SETNORM-1
FFF1:  7F                  DFB  #SETINV-1
FFF2:  5D                  DFB  #LIST-1
FFF3:  CC                  DFB  #WRITE-1
FFF4:  B5                  DFB  #GO-1
FFF5:  FC                  DFB  #READ-1
FFF6:  17                  DFB  #SETMODE-1
FFF7:  17                  DFB  #SETMODE-1
FFF8:  F5                  DFB  #CRMON-1
FFF9:  03                  DFB  #BLANK-1
FFFA:  FB                  DFB  #NMI          NMI VECTOR
FFFB:  03                  DFB  #NMI/256
FFFC:  59                  DFB  #RESET        RESET VECTOR
FFFD:  FF                  DFB  #RESET/256
FFFE:  86                  DFB  #IRQ          IRQ VECTOR
FFFF:  FA                  DFB  #IRQ/256
                XQTNZ      EQU  $3C
```

SYMBOL TABLE
(NUMERICAL ORDER)

0000	LOC0
0022	WNDTOP
0026	GBASL
002A	BAS2L
002D	V2
002E	FORMAT
0030	COLOR
0034	YSAV
0038	KSWL
003C	A1L
0040	A3L
0044	A5L
0047	YREG
004F	RNDH
03F2	SOFTEV
03FB	NMI
C000	IOADR
C030	SPKR
C053	MIXSET
C057	HIRES
C05B	CLRAN1
C05F	CLRAN3
CFFF	CLRROM
F80C	RTMASK
F826	VLINEZ
F836	CLRTOP
F856	GBCALC
F87F	RTMSKZ
F8A5	ERR
F8C9	MNNDX3
F8F5	NXTCOL
F926	PRADR3
F940	PRNTYX
F94A	PRBL2
F956	PCADJ3
F9A6	FMT2
FA00	MNEMR
FA62	RESET
FAA3	NOFIX
FABA	SLOOP
FAE4	RDSP1
FB11	XLTBL
FB2E	RTS2D
FB4B	SETWND
FB6F	SETPWRC
FB97	ESCOLD
FBD0	BASCLC2
FBF0	STORADV
FC10	BS
FC2B	RTS4
FC58	HOME
FC76	SCRL1
FC9E	CLEOLZ
FCAA	WAIT3
FCC9	HEADR
FCE5	WRTAPE
FCFD	RDBIT
FD2F	ESC
FD62	CANCEL
0001	LOC1
0023	WNDBTM
0027	GBASH
002B	BAS2H
002D	RMNEM
002F	LASTIN
0031	MODE
0035	YSAV1
0039	KSWH
003D	A1H
0041	A3H
0045	A5H
0048	STATUS
0095	PICK
03F4	PWREDUP
03FE	IRQLOC
C000	KBD
C050	TXTCLR
C054	LOWSCR
C058	SETAN0
C05C	SETAN2
C060	TAPEIN
E000	BASIC
F80E	PLOT1
F828	VLINE
F838	CLRSC2
F864	SETCOL
F882	INSDS1
F8A9	GETFMT
F8D0	INSTDSP
F8F9	PRMN2
F92A	PRADR4
F941	PRNTAX
F94C	PRBL3
F95C	PCADJ4
F9B4	CHAR1
FA40	IRQ
FA6F	INITAN
FAA6	PWRUP
FAC7	NXTBYT
FAFD	PWRCON
FB19	RTBL
FB2F	INIT
FB5B	TABV
FB78	VIDWAIT
FB9B	ESCNOW
FBD9	BELL1
FBF4	ADVANCE
FC1A	UP
FC2C	ESC1
FC62	CR
FC8C	SCRL2
FCA0	CLEOL2
FCB4	NXTA4
FCD6	WRBIT
FCEC	RDBYTE
FD0C	RDKEY
FD35	RDCHAR
FD67	GETLNZ
0020	WNDLFT
0024	CH
0028	BASL
002C	H2
002E	MASK
002F	LENGTH
0032	INVFLG
0036	CSWL
003A	PCL
003E	A2L
0042	A4L
0045	ACC
0049	SPNT
0200	IN
03F5	AMPERV
0400	LINE1
C010	KBDSTRB
C051	TXTSET
C055	HISCR
C059	CLRAN0
C05D	CLRAN2
C064	PADDL0
E003	BASIC2
F819	HLINE
F831	RTS1
F83C	CLRSC3
F871	SCRN
F88C	INSDS2
F8BE	MNNDX1
F8D4	PRNTOP
F910	PRADR1
F930	PRADR5
F944	PRNTX
F953	PCADJ
F961	RTS2

F9BA	CHAR2
FA4C	BREAK
FA81	NEWMON
FAA9	SETPG3
FAD7	REGDSP
FB02	DISKID
FB1E	PREAD
FB39	SETTXT
FB60	APPLEII
FB88	KBDWAIT
FBA5	ESCNEW
FBE4	BELL2
FBFC	RTS3
FC22	VTAB
FC42	CLREOP
FC66	LF
FC95	SCRL3
FCA8	WAIT
FCBA	NXTA1
FCDB	ZERDLY
FCEE	RDBYT2
FD1B	KEYIN
FD3D	NOTCR
FD6A	GETLN
0021	WNDWDTH
0025	CV
0029	BASH
002C	LMNEM
002E	CHKSUM
002F	SIGN
0033	PROMPT
0037	CSWH
003B	PCH
003F	A2H
0043	A4H
0046	XREG
004E	RNDL
03F0	BRKV
03F8	USRADR
07F8	MSLOT
C020	TAPEOUT
C052	MIXCLR
C056	LORES
C05A	SETAN1
C05E	SETAN3
C070	PTRIG
F800	PLOT
F81C	HLINE1
F832	CLRSCR
F847	GBASCALC
F879	SCRN2
F89B	IEVEN
F8C2	MNNDX2
F8DB	PRNTBL
F914	PRADR2
F938	RELADR
F948	PRBLNK
F954	PCADJ2
F962	FMT1
F9C0	MNEML
FA59	OLDBRK
FA9B	FIXSEV
FAAB	SETPLP
FADA	RGDSP1
FB09	TITLE
FB25	PREAD2
FB40	SETGR
FB65	STITLE
FB94	NOWAIT
FBC1	BASCALC
FBEF	RTS2B
FBFD	VIDOUT
FC24	VTABZ
FC46	CLEOP1
FC70	SCROLL
FC9C	CLREOL
FCA9	WAIT2
FCC8	RTS4B
FCE2	ONEDLY
FCFA	RD2BIT
FD21	KEYIN2
FD5F	NOTCR1
FD71	BCKSPC
FD75	NXTCHAR
FD92	PRA1
FDB3	XAM
FDD1	ADD
FDED	COUT
FE04	BLANK
FE1D	SETMDZ
FE36	VFY
FE75	A1PC
FE84	SETNORM
FE8D	INPRT
FE9B	IOPRT
FEB3	BASCONT
FEC4	STEPZ
FEED	WRBYTE
FF0A	RD2
FF3F	RESTORE
FF59	OLDRST
FF7A	CHRSRCH
FFA2	NXTBS2
FFC7	ZMODE
FD7E	CAPTST
FD96	PRYX2
FDB6	DATAOUT
FDDA	PRBYTE
FDF0	COUT1
FE0B	STOR
FE20	LT
FE58	VFYOK
FE78	A1PCLP
FE86	SETIFLG
FE93	SETVID
FEA7	IOPRT1
FEB6	GO
FECA	USR
FEEF	WRBYT2
FF16	RD3
FF44	RESTR1
FF65	MON
FF8A	DIG
FFA7	GETNUM
FFCC	CHRTBL
FD84	ADDINP
FDA3	XAM8
FDC5	RTS4C
FDE3	PRHEX
FDF6	COUTZ
FE17	RTS5
FE22	LT2
FE5E	LIST
FE7F	A1PCRTS
FE89	SETKBD
FE95	OUTPORT
FEA9	IOPRT2
FEBF	REGZ
FECD	WRITE
FEF6	CRMON
FF2D	PRERR
FF4A	SAVE
FF69	MONZ
FF90	NXTBIT
FFAD	NXTCHR
FFE3	SUBTBL
FD8E	CROUT
FDAD	MOD8CHK
FDC6	XAMPM
FDE5	PRHEXZ
FE00	BL1
FE18	SETMODE
FE2C	MOVE
FE63	LIST2
FE80	SETINV
FE8B	INPORT
FE97	OUTPRT
FEB0	XBASIC
FEC2	TRACE
FED4	WR1
FEFD	READ
FF3A	BELL

FF4C SAV1
FF73 NXTITM
FF98 NXTBAS
FFBE TOSUB

SYMBOL TABLE
(ALPHABETICAL ORDER)

003D A1H
FE7F A1PCRTS
0040 A3L
0044 A5L
FBF4 ADVANCE
002A BAS2L
0029 BASH
FD71 BCKSPC
FE00 BL1
FC10 BS
F9BA CHAR2
0024 CH
C059 CLRAN0
FC9C CLREOL
F83C CLRSC3
FDED COUT
FC62 CR
0025 CV
F8A5 ERR
FB97 ESCOLD
F9A6 FMT2
0026 GBASL
FD6A GETLN
FCC9 HEADR
F819 HLINE
0200 IN
F882 INSDS1
C000 IOADR
03FE IRQLOC
C000 KBD
0038 KSWL
0400 LINE1
0000 LOCO
FE22 LT2
C053 MIXSET
F8C2 MNNDX2
FF69 MONZ
FA81 NEWMON
FD5F NOTCR1
FF98 NXTBAS
FD75 NXTCHAR
FA59 OLDBRK
FE97 OUTPRT
F956 PCADJ3
0095 PICK
F910 PRADR1
F930 PRADR5
FDDA PRBYTE
FDE3 PRHEX
F8DB PRNTBL
0033 PROMPT
03F4 PWREDUP
FF16 RD3
FD35 RDCHAR
FAD7 REGDSP
FF3F RESTORE
004F RNDH
F87F RTMSKZ
F961 RTS2
003C A1L
003F A2H
0043 A4H
0045 ACC
03F5 AMPERV
FBC1 BASCALC
E000 BASIC
FBD9 BELL1
FE04 BLANK
FD62 CANCEL
002E CHKSUM
FCA0 CLEOL2
C05B CLRAN1
FC42 CLREOP
F832 CLRSCR
FDF0 COUT1
FEF6 CRMON
FDB6 DATAOUT
FC2C ESC1
FD2F ESC
002E FORMAT
F856 GBCALC
FFA7 GETNUM
C057 HIRES
FC58 HOME
FB2F INIT
F88C INSDS2
FEA7 IOPRT1
FA40 IRQ
FD1B KEYIN
002F LASTIN
FE5E LIST
0001 LOC1
FE20 LT
F9C0 MNEML
FBC9 MNNDX3
FF65 MON
03FB NMI
FB94 NOWAIT
FF90 NXTBIT
FFAD NXTCHR
FF59 OLDRST
C064 PADDL0
F95C PCADJ4
F80E PLOT1
F914 PRADR2
F94A PRBL2
FB1E PREAD
FDE5 PRHEXZ
F8D4 PRNTOP
FD96 PRYX2
FAA6 PWRUP
FCFD RDBIT
FD0C RDKEY
FEBF REGZ
FF44 RESTR1
004E RNDL
F831 RTS1
FBFC RTS3
FE78 A1PCLP
003E A2L
0042 A4L
FD84 ADDINP
FB60 APPLEII
FBD0 BASCLC2
E003 BASIC2
FBE4 BELL2
FA4C BREAK
FD7E CAPTST
FF7A CHRSRCH

FC9E	CLEOLZ
C05D	CLRAN2
CFFF	CLRROM
F836	CLRTOP
FDF6	COUTZ
0037	CSWH
FF8A	DIG
FBA5	ESCNEW
FA9B	FIXSEV
F847	GBASCALC
F8A9	GETFMT
FEB6	GO
C055	HISCR
F89B	IEVEN
FE8B	INPORT
F8D0	INSTDSP
FEA9	IOPRT2
C010	KBDSTRB
FD21	KEYIN2
002F	LENGTH
FE63	LIST2
C056	LORES
002E	MASK
FA00	MNEMR
FDAD	MOD8CHK
FE2C	MOVE
FAA3	NOFIX
FCBA	NXTA1
FFA2	NXTBS2
F8F5	NXTCOL
FCE2	ONEDLY
F954	PCADJ2
003B	PCH
F800	PLOT
F926	PRADR3
F94C	PRBL3
FB25	PREAD2
F8F9	PRMN2
F944	PRNTX
C070	PTRIG
FCFA	RD2BIT
FCEE	RDBYT2
FAE4	RDSP1
F938	RELADR
FADA	RGDSP1
FB19	RTBL
FBEF	RTS2B
FCC8	RTS4B
FE75	A1PC
0041	A3H
0045	A5H
FDD1	ADD
002B	BAS2H
FEB3	BASCONT
0028	BASL
FF3A	BELL
03F0	BRKV
F9B4	CHAR1
FFCC	CHRTBL
FC46	CLEOP1
C05F	CLRAN3
F838	CLRSC2
0030	COLOR
FD8E	CROUT
0036	CSWL
FB02	DISKID
FB9B	ESCNOW
F962	FMT1
0027	GBASH
FD67	GETLNZ
002C	H2
F81C	HLINE1
FA6F	INITAN
FE8D	INPRT
0032	INVFLG
FE9B	IOPRT
FB88	KBDWAIT
0039	KSWH
FC66	LF
002C	LMNEM
C054	LOWSCR
C052	MIXCLR
F8BE	MNNDX1
0031	MODE
07F8	MSLOT
FD3D	NOTCR
FCB4	NXTA4
FAC7	NXTBYT
FF73	NXTITM
FE95	OUTPORT
F953	PCADJ
003A	PCL
FD92	PRA1
F92A	PRADR4
F948	PRBLNK
FF2D	PRERR
F941	PRNTAX
F940	PRNTYX
FAFD	PWRCON
FF0A	RD2
FCEC	RDBYTE
FEFD	READ
FA62	RESET
002D	RMNEM
F80C	RTMASK
FB2E	RTS2D
FDC5	RTS4C
FE17	RTS5
FC2B	RTS4
FC76	SCRL1
F879	SCRN2
C05C	SETAN2
FE86	SETIFLG
FE18	SETMODE
FB6F	SETPWRC
002F	SIGN
0049	SPNT
FE0B	STOR
C060	TAPEIN
FEC2	TRACE
FECA	USR
FE58	VFYOK
F828	VLINE
FCA8	WAIT
0022	WNDTOP
FEEF	WRBYT2
FDA3	XAM8
FB11	XLTBL
0034	YSAV
FC8C	SCRL2
FC70	SCROLL
C05E	SETAN3
FE80	SETINV
FE84	SETNORM
FB39	SETTXT
FABA	SLOOP
0048	STATUS
FBF0	STORADV
C020	TAPEOUT
C050	TXTCLR
03F8	USRADR
FBFD	VIDOUT
FC24	VTABZ
FCAA	WAIT3
0021	WNDWDTH
FEED	WRBYTE
FDC6	XAMPM
0046	XREG
FCDB	ZERDLY
FF4C	SAV1
FC95	SCRL3
C058	SETAN0
F864	SETCOL
FE89	SETKBD
FAA9	SETPG3
FE93	SETVID
03F2	SOFTEV
FEC4	STEPZ
FFE3	SUBTBL
FB09	TITLE
C051	TXTSET
002D	V2
FB78	VIDWAIT
FC22	VTAB
0023	WNDBTM
FED4	WR1

```
FECD WRITE
FDB3 XAM
0047 YREG
FFC7 ZMODE
FF4A SAVE
F871 SCRN
C05A SETAN1
FB40 SETGR
FE1D SETMDZ
FAAB SETPLP
FB4B SETWND
C030 SPKR
FB65 STITLE
FB5B TABV
FFBE TOSUB
FC1A UP
FE36 VFY
F826 VLINEZ
FCA9 WAIT2
0020 WNDLFT
FCD6 WRBIT
FCE5 WRTAPE
FEB0 XBASIC
0035 YSAV1

SYMBOL TABLE SIZE
2589   BYTES USED
2531   BYTES REMAINING

SLIST 4A
```

附　　錄　　D

GETTING APPLESOFT BASIC UP AND RUNNING

Getting APPLESOFT BASIC Up And Running

APPLE Computer Inc. offers two versions of the BASIC programming language. Integer BASIC, described in the APPLE II BASIC Programming Manual, is a very fast BASIC suited for many applications, especially in education, game playing, and graphics. The other version of BASIC is called "APPLESOFT" and is better suited for most business and scientific applications.

APPLESOFT BASIC is available in two versions. Firmware APPLESOFT comes with APPLESOFT in ROM on a printed circuit card (APPLE Part Number A2BØØØ9X) which plugs directly into the APPLE II. With this option, the flick of a switch and two key-strokes start the APPLE II running in APPLESOFT. Aside from this convenience, having APPLESOFT in ROM saves about 1ØK of memory and saves much time loading the language in at every use, from a cassette tape. The main body of this manual assumes you have the firmware APPLESOFT card. If you are using the cassette version of APPLESOFT, see PART II of this appendix for special instructions and notes on where your APPLESOFT differs from that described in the rest of this manual.

Note: in this manual, the word reset means to press the key marked RESET, return means to press the key marked RETURN, and ctrl B means to type B while holding down the key marked CTRL .

AN IMPORTANT NOTE:
One of the functions of the prompt character, besides PROMPTing you for input to the computer, is to identify at a glance which language the computer is programmed to respond to at that time. For instance, up till now you have seen two prompt characters:

*   for the Monitor program (when you press RESET)

>   for APPLE Integer BASIC (the normal integer BASIC)

Now we introduce a third prompt character:

]   for APPLESOFT floating-point BASIC.

By simply looking at this prompt character, you can easily tell (if you forget) which language the computer is in.

FIRMWARE APPLESOFT

INSTALLING THE FIRMWARE APPLESOFT BOARD

The firmware APPLESOFT card simply plugs into a socket inside the APPLE II. Care must be exercised, however, so follow these instructions exactly:

1) Turn the APPLE off: very important to prevent damaging the computer.

2) Remove the cover from the APPLE II. This is done by pulling up on the cover at the rear edge (the edge farthest from the keyboard) until the two corner fasteners pop apart. Do not continue to lift the rear edge, but slide the cover backward until it comes free.

3) Inside the APPLE II, across the rear of the circuit board, there is a row of eight long, narrow sockets called "slots." The leftmost one (looking at the computer from the keyboard end) is slot #Ø; the rightmost one is slot #7. Hold the APPLESOFT card so that its switch is toward the back of the computer; insert the "fingers" portion of the card into the leftmost slot, slot #Ø. The fingers will enter the slot with some friction, and will then seat firmly. The APPLESOFT card must be placed in slot #Ø.

4) The switch on the back of the APPLESOFT card should protrude part.way through the slot on the back of the APPLE II.

5) Replace the APPLE's cover: first slide the front edge into place, then press down on the two rear corners until they pop into place.

7) Now turn on the APPLE II.

USING THE FIRMWARE APPLESOFT BOARD

With the APPLESOFT card's switch in the downward position, the APPLE II will begin operating in Integer BASIC when you use reset ctrl B (this manual's way of saying: press the key marked RESET, then hold down the key marked CTRL while typing B). You will see the prompt character >, which indicates Integer BASIC.

With the switch in the upward position, reset ctrl B will bring up APPLESOFT BASIC, instead of Integer BASIC. The prompt character] tells you you're in APPLESOFT.

When using the Disk Operating System, the computer will automatically choose Integer BASIC or APPLESOFT, as required. It does not matter in which position the switch is set.

You can also change from Integer BASIC to APPLESOFT, or vice versa, without operating the switch on the firmware card. To put the computer into APPLESOFT, use

reset CØ8Ø return
ctrl B return

and to put the computer into Integer BASIC, use

reset CØ81 return
ctrl B return

ANOTHER IMPORTANT NOTE:
Sometime you may accidently hit RESET and find yourself in the Monitor, as shown by the * prompt character. You may be able to return to APPLESOFT BASIC, with APPLESOFT and your program intact, by typing

ctrl C return

附　　錄　　E

名　　詞　　解　　釋

6502: The manufacturer's name for the microprocessor at the heart of your Apple.

Address: As a noun: the particular number associated with each memory location. On the Apple, an address is a number between 0 and 65535 (or $0000 and $FFFF hexadecimal). As a verb: to refer to a particular memory location.

Address Bus: The set of wires, or the signal on those wires, which carry the binary-encoded address from the microprocessor to the rest of the computer.

Addressing mode: The Apple's 6502 microprocessor has thirteen distinct ways of referring to most locations in memory. These thirteen methods of forming addresses are called **addressing modes**.

Analog: Analog measurements, as opposed to digital measurements, use an continuously variable physical quantity (such as length, voltage, or resistance) to represent values. Digital measurements use precise, limited quantities (such as presence or absence of voltages or magnetic fields) to represent values.

AND: A binary function which is "on" if and only if all of its inputs are "cn".

Apple: 1. The round fleshy fruit of a Rosaceous tree (Pyrus Malus). 2. A brand of personal computer. 3) Apple Computer, Inc., manufacturer of home and personal computers.

ASCII: An acronym for the American Standard Code for Information Interchange (often called "USASCII" or misinterpreted as "ASC-II") This standard *code* assigns a unique value from 0 to 127 to each of 128 numbers, letters, special characters, and control characters.

Assembler: 1) One who assembes electronic or mechanical equipment. 2) A program which converts the *mnemonics* and *symbols* of assembly language into the *opcodes* and *operands* of machine language.

Assembly language: A language similar in structure to machine language, but made up of *mnemonics* and *symbols.* Programs written in assembly language are slightly less difficult to write and understand than programs in machine language.

BASIC: Acronym for "Beginner's All-Purpose Symbolic Instruction Code". BASIC is a *higher-level language*, similar in structure to FORTRAN but somewhat easier to learn. It was invented by Kemney and Kurtz at Dartmouth College in 1963 and has proved to be the most popular language for personal computers.

Binary: A number system with two digits, "0" and "1", with each digit in a binary number representing a power of two. Most digital computers are binary, deep down inside. A binary signal is easily expressed by the presence or absence of something, such as an electrical potential or a magnetic field.

Binary Function: An operation performed by an electronic circuit which has one or more inputs and only one output. All inputs and outputs are binary signals. See *AND OR,* and *Exclusive-OR.*

Bit: A *Binary digIT.* The smallest amount of information which a computer can hold. A single bit specifies a single value: "0" or "1". Bits can be grouped to form larger values (see *Byte* and *Nybble*).

Board: See *Printed Circuit Board.*

Bootstrap ("boot"): To get a system running from a *cold-start.* The name comes from the machine's attempts to "pull itsef off the ground by tugging on its own bootstraps."

Buffer: A device or area of memory which is used to hold something temporarily. The "picture buffer" contains graphic information to be displayed on the video screen; the "input buffer" holds a partially formed input line.

Bug: An error. A *hardware bug* is a physical or electrical malfunction or design error. A *software* bug is an error in programming, either in the logic of the program or typographical in nature. See "feature".

Bus: A set of wires or *traces* in a computer which carry a related set of data from one place to another, or the data which is on such a bus.

Byte: A basic unit of measure of a computer's memory. A byte usualy comprises eight *bits.* Thus, it can have a value from Ø to 255. Each character in the *ASCII* can be represented in one byte. The Apple's memory locations are all one byte, and the Apple's addresses of these locations consist of two bytes.

Call: As a verb: to leave the program or subroutine which is currently executing and to begin another, usualy with the intent to return to the original program or subroutine. As a noun: an instruction which calls a subroutine.

Character: Any *graphic* symbol which has a specific meaning to people. Letters (both upper- and lower-case), numbers, and various symbols (such as punctuation marks) are all characters.

Chip: See *Integrated Circuit.*

Code: A method of representing something in terms of something else. The ASCII code represents characters as binary numbers, the BASIC language represents algorithms in terms of program statements. **Code** is also used to refer to programs, usually in *low-level languages.*

Cold-start: To begin to operate a computer which has just been turned on.

Color burst: A signal which color television sets recognize and convert to the colored dots you see on a color TV screen. Without the color burst signal, all pictures would be black-and-white.

Computer: Any device which can recieve and store a set of *instructions,* and then act upon those instructions in a predetermined and predictable fashion. The definition implies that both the instruction and the *data* upon which the instructions act can be changed. A device whose instructions cannot be changed is not a computer.

Control (CTRL) character: Characters in the *ASCII* character set which usually have no graphic representation, but are used to control various functions. For example, the RETURN control character is a signal to the Apple that you have finished typing an *input line* and you wish the computer to act upon it.

CRT: Acronym for "Cathode-Ray Tube", meaning any television screen, or a device containing such a screen.

Cursor: A special symbol which reminds you of a certain position on something. The cursor on a slide rule lets you line up numbers; the cursor on the Apple's screen reminds you of where you are when you are typing.

Data (datum): Information of any type.

Debug: To find *bugs* and eliminate them.

DIP: Acronym for "Dual In-line Package", the most common container for an Integrated Circuit. DIPs have two parallel rows of *pins*, spaced on one-tenth of an inch centers. DIPs usually come in 14-, 16-, 18-, 20-, 24-, and 40-pin configurations.

Disassembler: A program which converts the *opcodes* of *machine language* to the *mnemonics* of *assembly language*. The opposite of an *assembler*.

Display: As a noun: any sort of output device for a computer, usually a *video* screen. As a noun: to place information on such a screen.

Edge connector: A socket which mates with the edge of a *printed circuit board* in order to exchange electrical signals.

Entry point: The location used by a machine-language subroutine which contains the first executable instruction in that subroutine; consequently, often the beginning of the subroutine.

Excusive-OR: A binary function whose value is "off" only if all of its inputs are "off", or all of its inputs are "on".

Execute: To perform the intention of a command or instruction. Also, to run a program or a portion of a program.

Feature: A *bug* as described by the marketing department.

Format: As a noun: the physical form in which something appears. As a verb: to specify such a form.

Graphic: Visible as a distinct, recognizable shape or color.

Graphics: A system to display graphic items or a collection of such items.

Hardware: The physical parts of a computer.

Hexadecimal: A number system which uses the ten digits Ø through 9 and the six letters A through F to represent values in base 16. Each hexadecimal digit in a hexadecimal number represents a power of 16. In this manual, all hexadecimal numbers are preceded by a dollar sign ($).

High-level Language: A *language* which is more intelligible to humans than it is to machines.

High-order: The most important, or item with the highest vaue, of a set of similar items. The high-order bit of a byte is that which has the highest place value.

High part: The *high-order* byte of a two-byte address. In decimal, the high part of an address is the quotient of the address divided by 256. In the 65Ø2, as in many other microprocessors, the high part of an address comes last when that address is stored in memory.

Hz (Hertz): Cycles per second. A bicycle wheel which makes two revolutions in one second is running at 2Hz. The Apple's microprocessor runs at 1,023,000Hz.

I/O: See *Input/Output.*

IC: See *Integrated Circuit.*

Input: As a noun: data which flows from the outside world into the computer. As a verb: to obtain data from the outside world.

Input/Output (I/O): The software or hardware which exchanges data with the outside word.

Instruction: The smallest portion of a program that a computer can execute. In 65Ø2 machine language, an instruction comprises one, two, or three bytes; in a higher-level language, instructions may be many characters long.

Integrated circuit: A small (less than the size of a fingernail and about as thin) wafer of a glassy material (usually silicon) into which has been etched an electronic circuit. A single IC can contain from ten to ten thousand discrete electronic components. ICs are usually housed in *DIPs* (see above), and the term IC is sometimes used to refer to both the circuit and its package.

Interface: An exchange of information between one thing and another, or the mechanisms which make such an exchange possible.

Interpreter: A program, usualy written in machine language, which understands and executes a higher-level language.

Interrupt: A physical effect which causes the computer to jump to a special interrupt-handling subroutine. When the interrupt has been taken care of, the computer resumes execution of the interrupted program with no noticeable change. Interrupts are used to signal the computer that a particular device wants attention.

K: Stands for the greek prefix "Kilo", meaning one thousand. In common computer-reated usage, "K" usually represents the quantity $2^{1Ø}$, or 1Ø24 (hexadecimal $4ØØ).

Kilobyte: 1,024 bytes.

Language: A computer language is a code which (hopefully!) both a programmer and his computer understand. The programmer expresses what he wants to do in this code, and the computer understands the code and performs the desired actions.

Line: On a video screen, a "line" is a horizontal sequence of graphic symbols extending from one edge of the screen to the other. To the Apple, an *input line* is a sequence of up to 254 characters, terminated by the control character RETURN. In most places which do not have personal computers, a line is something you wait in to use the computer.

Low-level Language: A *language* which is more intelligible to machines than it is to humans.

Low-order: The least important, or item with the least vaue, of a set of items. The low-order bit in a byte is the bit with the least place vaue.

Low part: The *low-order* byte of a two-byte address. In decimal, the low part of an address is the remainder of the address divided by 256, also called the "address *modulo* 256." In the 65Ø2, as in many other microprocessors, the low part of an address comes first when that address is stored in memory.

Machine language: The lowest level language which a computer understands. Machine

languages are usually binary in nature. Instructions in machine language are single-byte *opcodes* sometimes followed by various *operands.*

Memory address: A memory address is a two-byte value which selects a single memory location out of the *memory map.* Memory addresses in the Apple are stored with their low-order bytes first, followed by their high-order bytes.

Memory location: The smallest subdivision of the memory map to which the computer can refer. Each memory location has associated with it a unique *address* and a certain *value.* Memory locations on the Apple comprise one byte each.

Memory Map: This term is used to refer to the set of all memory locations which the microprocesor can address directly. It is also used to describe a graphic representation of a system's memory.

Microcomputer: A term used to described a computer which is based upon a microprocessor.

Microprocessor: An integrated circuit which understands and executes machine language programs.

Mnemonic: An acronym (or any other symbol) used in the place of something more difficut to remember. In *Assembly Language,* each machine language opcode is given a three letter mnemonic (for example, the opcode $60 is given the mnemonic RTS, meaning "ReTurn from Subroutine").

Mode: A condition or set of conditions under which a certain set of rules apply.

Modulo: An arithmetic function with two operands. *Modulo* takes the first operand, divides it by the second, and returns the remainder of the division.

Monitor: 1) A closed-circuit television receiver. 2) A program which allows you to use your computer at a very low level, often with the values and addresses of individual memory locations.

Multiplexer: An electronic circuit which has many data inputs, a few selector inputs, and one output. A multiplexer connects one of its many data inputs to its output. The data input it chooses to connect to the output is determined by the selector inputs.

Mux: See *Multiplexer.*

Nybble: Colloquial term for half of a byte, or four bits.

Opcode: A machine language instruction, numerical (often binary) in nature.

OR: A binary function whose value is "on" if at least one of its inputs are "on".

Output: As a noun, data generated by the computer whose destination is the real world. As a verb, the process of generating or transmitting such data.

Page: 1) A screenfull of information on a video display. 2) A quantity of memory locations, addressible with one byte. On the Apple, a "page" of memory contains 256 locations.

Pascal: A noted French scientist.

PC board: See *Printed Circuit Board.*

Peripheral: Something attached to the computer which is not part of the computer itself. Most peripherals are input and/or output devices.

Personal Computer: A computer with *memory*, *languages*, and *peripherals* which are well-suited for use in a home, office, or school.

Pinout: A description of the function of each pin on an IC, often presented in the form of a diagram.

Potentiometer: An electronic component whose resistance to the flow of electrons is proportional to the setting of a dial or knob. Also known as a "pot" or "variable resistor".

Printed Circuit Board: A sheet of fiberglass or epoxy onto which a thin layer of metal has been applied, then etched away to form *traces*. Electronic components can then be attatched to the board with molten solder, and they can exchange electronic signals via the etched traces on the board. Small printed circuit boards are often called "cards", especially if they are meant to connect with *edge connectors*.

Program: A sequence of instructions which describes a process.

PROM: Acronym for "*Programmable Read-Only Memory*". A PROM is a ROM whose contents can be altered by electrical means. Information in PROMs does not disappear when the power is turned off. Some PROMs can be erased by ultraviolet light and be reprogrammed.

RAM: See *Random-Access Memory*.

Random-Access Memory (RAM): This is the main memory of a computer. The acronym RAM can be used to refer either to the integrated circuits which make up this type of memory or the memory itself. The computer can store values in distinct locations in RAM and recall them again, or alter and re-store them if it wishes. On the Apple, as with most small computers, the values which are in RAM memory are lost when the power to the computer is turned off.

Read-Only Memory (ROM): This type of memory is usually used to hold important programs or data which must be available to the computer when the power is first turned on. Information in ROMs is placed there in the process of manufacturing the ROMs and is unalterable. Information stored in ROMs does not disappear when the power is turned off.

Reference: 1) A source of information, such as this manual. 2) As a verb, the action of examining or altering the contents of a memory location. As a noun, such an action.

Return: To exit a subroutine and go back to the program which called it.

ROM: See *Read-Only Memory*.

Run: To follow the sequence of instructions which comprise a program, and to complete the process outlined by the instructions.

Scan line: A single sweep of a cathode beam across the face of a *cathode-ray tube*.

Schematic: A diagram which represents the electrical interconnections and circuitry of an electronic device.

Scroll: To move all the text on a display (usually upwards) to make room for more (usually at the bottom).

Soft switch: A two-position switch which can be "thrown" either way by the software of a computer.

Software: The *programs* which give the hardware something to do.

Stack: A reserved area in memory which can be used to store information temporarily. The information in a stack is referenced not by address, but in the order in which it was placed on the stack. The last datum which was "pushed" onto the stack will be the first one to be "popped" off it.

Strobe: A momentary signal which indicates the occurrence of a specific event.

Subroutine: A segment of a program which can be executed by a single *call.* Subroutines are used to perform the same sequence of instructions at many different places in one program.

Syntax: The structure of instructions in a given *language.* If you make a mistake in entering an instruction and garble the syntax, the computer sometimes calls this a "SYNTAX ERROR."

Text: Characters, usually letters and numbers. "Text" usually refers to large chunks of English, rather than computer, language.

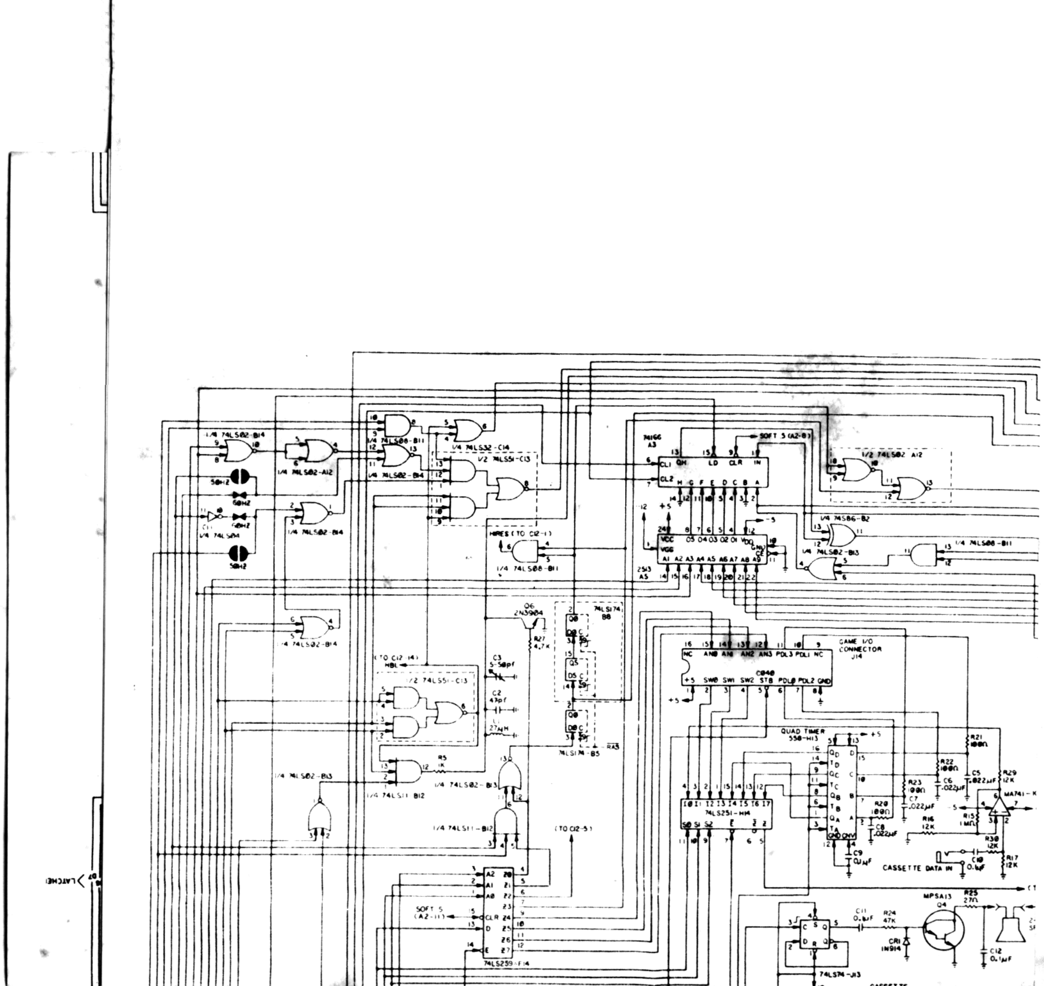
Toggle switch: A two-position switch which can only flip from one position to the other and back again, and cannot be directly set either way.

Trace: An etched conductive path on a *Printed-Circuit Board* which serves to electronically connect components.

Video: 1) Anything visual. 2) Information presented on the face of a *cathode-ray tube.*

Warm-start: To restart the operation of a computer after you have lost control of its language or operating system.

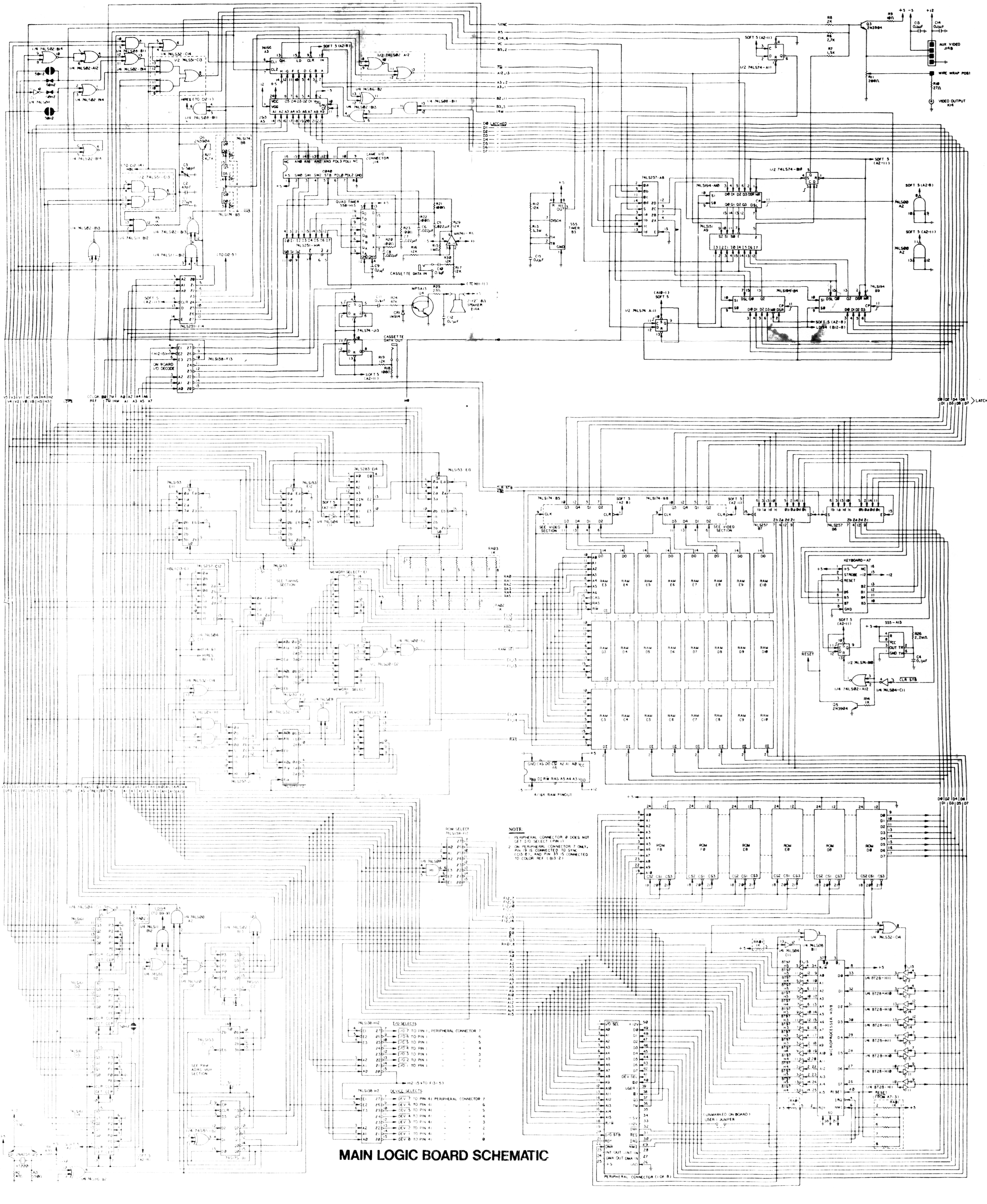
Window: Something out of which you jump when the power fails and you lose a large program. Really: a reserved area on a *display* which is dedicated to some special purpose.


GAME I/O CONNECTOR J14
QUAD TIMER 558-H13
CASSETTE DATA IN
74LS251-H4
74LS259-F14
MPSA13 Q4
Q6 2N3904
74LS174 B8
2513 A5
74166 A3
C040
SOFT 5 (A2-8)
HIRES (TO C12-1)
74LS74-J13
CASSETTE

Published & Printed
in Hong Kong
H. K. $ 25.00

APPLE II 系統介紹與使用手册

電腦技術社印行



MAIN LOGIC BOARD SCHEMATIC